VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ZVUKOVÝ MODUL PRO PLATFORMU FITKIT

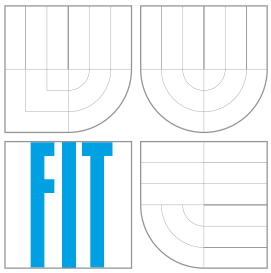
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL BARTOŠ

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ZVUKOVÝ MODUL PRO PLATFORMU FITKIT

SOUND MODULE FOR FITKIT PLATFORM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL BARTOŠ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÁCLAV ŠIMEK

BRNO 2009

Abstrakt

Tato práce se zabývá popisem modulu FITkitu, který umožňuje přehrávání zvukových souborů (mp3, ogg...). Dále rozšiřuje periferie FITkitu o barevný dotykový LCD diplej a USB konektor, který umožňuje připojení flash paměti.

Abstract

This work deals with module of the FITkit platform, which makes it able to play sound files like mp3, ogg, etc. The module also adds to FITkit some new peripherals: color LCD display with touch screen and USB interface, by which we can connect flash drive.

Klíčová slova

mp3, ogg, audio, zvuk, kodek, FITkit, modul, PIC, Microchip, LCD, dotykový displej, USB, flash, DPS, plošný spoj, vestavěný systém, grafická knihovna

Keywords

mp3, ogg, audio, sound, codec, FITkit, module, PIC, Microchip, LCD, display, touch screen, USB, flash, PCB, printed circuit, embedded system, graphics library

Citace

Pavel Bartoš: Zvukový modul pro platformu FITkit, diplomová práce, Brno, FIT VUT v Brně, 2009

Zvukový modul pro platformu FITkit

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Václava Šimka.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Pavel Bartoš
25. května 2009

Poděkování

Tímto děkuji Ing. Václavu Šimkovi za pomoc a cenné rady, které mi poskytl při vývoji této diplomové práce.

Zadání diplomové práce/6439/2008/xbarto41

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2008/2009

Zadání diplomové práce

Řešitel: **Bartoš Pavel, Bc.**
Obor: Počítačové systémy a sítě
Téma: **Zvukový modul pro platformu FITkit**
Kategorie: Počítačová architektura

Pokyny:

1. Seznamte se s výukovou platformou FITkit a jazykem VHDL pro návrh číslicových systémů.
2. Podrobně prostudujte strukturu a vlastnosti zvukového formátu MP3. Zaměřte se především na způsob dekódování.
3. Zvolte vhodný obvod s hardwarovou podporou procesu dekódování. Na jeho základě pak proveďte návrh modulu pro platformu FITkit, který umožní přehrávání souborů MP3.
4. V návrhovém systému Eagle nebo KiCAD vytvořte pro uvažovaný modul desku plošných spojů.
5. V jazyce VHDL implementujte řadič zajišťující načítání zvukových souborů MP3 z hostitelského PC nebo z dostupné paměti FLASH.
6. Funkčnost navrženého řešení bude demonstrována přehráním ukázkového souboru MP3.
7. Diskutujte možnosti dalšího rozšíření.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části diplomového projektu je požadováno:

- Splnění bodů 1-3 zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Šimek Václav, Ing.**, UPSY FIT VUT
Datum zadání: 22. září 2008
Datum odevzdání: 26. května 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

Obsah

Úvod

MP3 přehrávač je typickým vestavěným systémem. Vestavěný systém je jednoúčelový systém, ve kterém je řídicí počítač zcela zabudován do zařízení, které ovládá. Vzhledem k tomu, že systém je vytvořen k plnění konkrétních úkolů, lze jej ve značné míře optimalizovat a snížit tak cenu výrobku.

Cílem mé práce bylo vytvořit modul k platformě FITkit, který umožní přehrávat MP3 soubory. K tomu jsem využil hardwarový kodek, který je schopen dekódovat nejpoužívanější audio formáty. Modul jsem se rozhodl doplnit o barevný dotykový LCD displej, pevný disk a USB rozhraní.

V první kapitole se věnuji platformě FITkit. Jsou v ní popsány součásti FITkitu a programovatelné hradlové pole FPGA. Tuto kapitolu jsem částečně převzal ze své bakalářské práce[2], ve které jsem také pracoval s FITkitem.

V další kapitole je podrobně popsán zvukový formát MP3. Je zde rozebrán proces kódování i dekódování a popsána struktura souboru. Jsou zde stručně popsány i ostatní podporované zvukové formáty.

V kapitole č. 3 se zabývám návrhem hardware modulu. Nachází se zde popis výběru součástek, návrh modulu a desky plošných spojů i popis oživení.

Ve čtvrté kapitole popisuji konfiguraci FPGA, ve kterém se nachází řadič pevného disku a převodník rozhraní SPI na rozhraní UART.

V páté kapitole je popsán vývoj softwaru pro mikrokontrolér na modulu. Kapitolu lze rozdělit na část týkající se audio kodeku, část týkající se displeje a část týkající se načítání dat z USB paměti.

V poslední kapitole jsou popsány možnosti rozšíření mého řešení.

V rámci semestrálního projektu jsem se seznámil s platformou FITkit a formátem MP3, což je popsáno v kapitolách 1 a 2. Dále jsem v rámci semestrálního projektu vybral zvukový kodek a to je pospáno v třetí kapitole.

Kapitola 1

Platforma FITkit

FITkit je platforma obsahující mikrokontrolér, programovatelné hradlové pole, 16-ti znakový jednořádkový LCD displej, maticovou klávesnici, paměť RAM, audio zesilovač, převodník USB – USART a konektory pro připojení periferií. FITkit se připojuje k počítači přes USB rozhraní, které zároveň zajišťuje přívod napájecího napětí 5 V. Po připojení externího zdroje napájecího napětí je možné s FITkitem pracovat i bez počítače.

Obrázek 1.1: FITkit

Program pro mikrokontrolér se tvoří v jazyce C a překládá se pomocí GNU překladače. Do flash paměti mikrokontroléru se přeložený program nahrává z počítače přes USB rozhraní.

Architektura pro FPGA se popisuje jazykem VHDL. Nahrání konfigurace do FPGA je možné jen prostřednictvím mikroprocesoru. Konfiguraci je možné nahrát buď do flash paměti, nebo přímo do FPGA. V prvním případě je konfigurace do FPGA nahrána z flash paměti při každém resetu mikroprocesoru. V druhém případě zůstane flash paměť nedotčena, ale je třeba počítat s tím, že při resetu procesoru bude do FPGA nahrána konfigurace uložená v paměti flash.

1.1 Mikrokontrolér

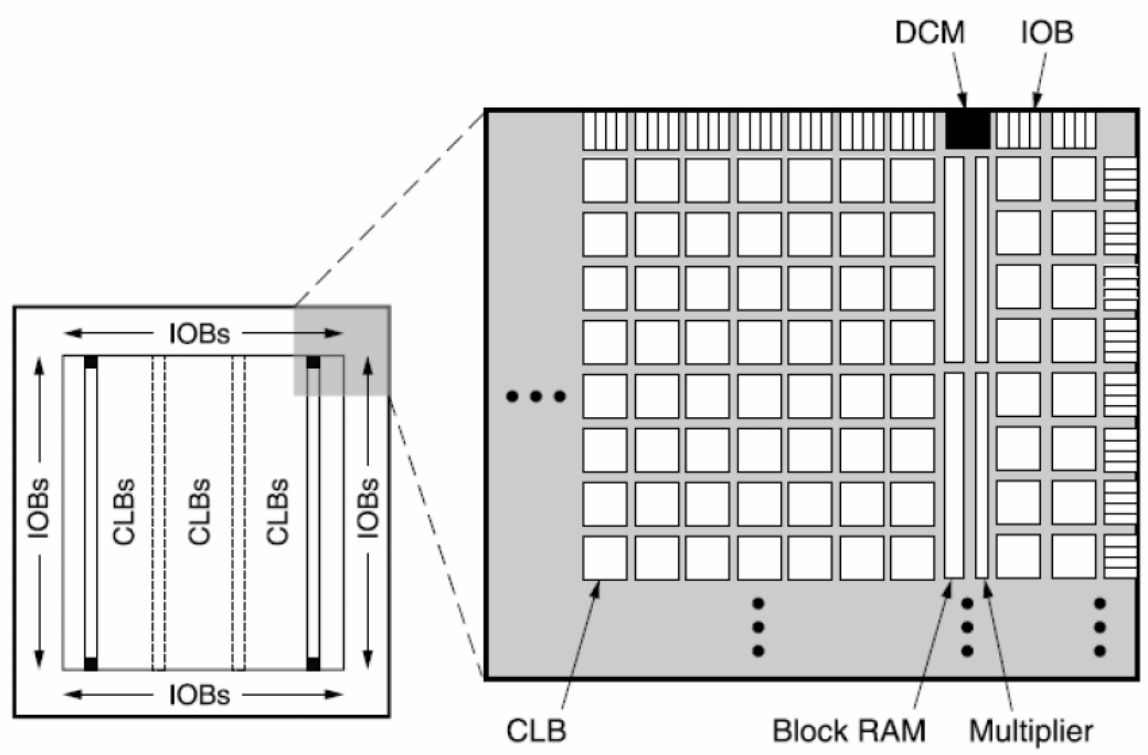
FITkit obsahuje nízkopříkonový mikrokontrolér MSP430F168 od firmy Texas Instruments. V této práci jej ale nijak nevyužívám, takže jej ani nebudu blíže popisovat.

1.2 Programovatelné hradlové pole[8, 11]

Programovatelné hradlové pole umístěné na FITkitu je řady Spartan 3 od firmy Xilinx. Tento reprogramovatelný hardware lze neomezeně modifikovat.

Všechna FPGA firmy Xilinx se konfigurují pomocí statické paměti RAM. To znamená, že po připojení napájecího napětí je nutné vždy znovu nahrát konfiguraci do FPGA. Výhodou tohoto řešení je téměř nekonečná reprogramovatelnost FPGA a také vysoká rychlost. Konfigurační propojky pracující na principu paměti RAM jsou totiž rychlejší než přepínače založené na principu EEPROM.

Interní struktura obvodu Spartan 3 je znázorněna na obrázku 1.2.

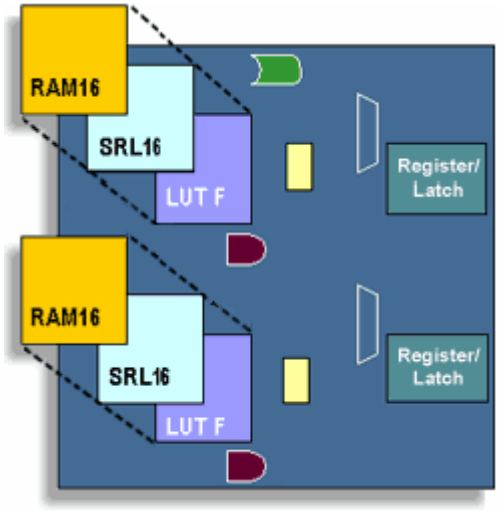


Obrázek 1.2: Vnitřní struktura FPGA Spartan 3

Obvod obsahuje vstupně-výstupní buňky (IOB), konfigurovatelné logické bloky (CLB), blokovou paměť RAM (Block RAM), násobičky a obvody pro řízení hodinového signálu (DCM).

Každý pin (buňka IOB) FPGA může být konfigurován jako vstup, výstup nebo obojí. Dva sousední piny lze zapojit jako diferenciální pár. Podporováno je velké množství standardů (LVTTL, LVCMOS, HSTTL, PCI-X, AGP a další).

Každý konfigurovatelný logický blok (CLB) obsahuje čtyři menší logické elementy (slice) a 2 nezávislé *carry* řetězce pro konstrukci rychlých sčítaček.



Obrázek 1.3: Vnitřní struktura slice

Slice (viz obr. 1.3) obsahuje 2 funkční generátory, které mohou být použity jako logické hradlo, nebo jako paměť o velikosti 16x1 bitů, a nebo jako posuvný registr. Slice ještě obsahuje 2 registry, multiplexory a *carry* logiku. Pomocí multiplexorů lze například vytvořit paměť o libovolné šířce dat.

Kapitola 2

Formáty zvukových souborů

2.1 Formát MP3

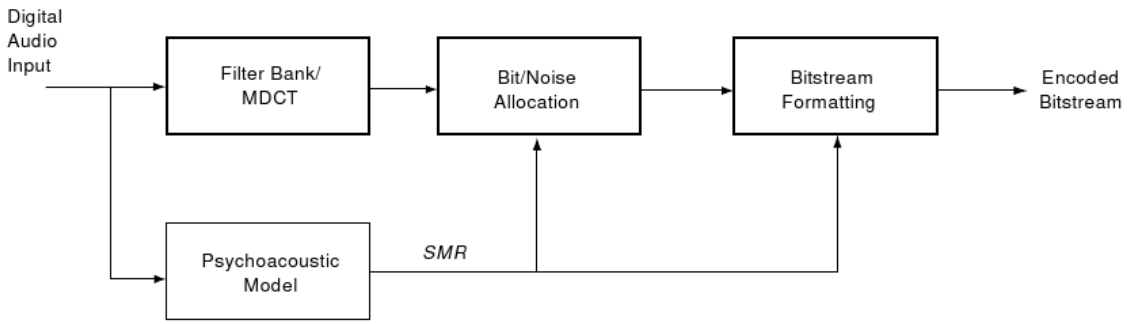
MP3 (MPEG-1[15] Layer 3) je formát ztrátové komprese zvukových souborů, založený na kompresním algoritmu MPEG (Motion Picture Experts Group). Při zachování vysoké kvality umožňuje zmenšit velikost hudebních souborů v CD kvalitě přibližně na desetinu, u mluveného slova však dává výrazně horší výsledky[14].

MP3 funguje tak, že za pomoci psychoakustického modelu odstraňuje ty části zvuku, které není většina lidí schopna rozpoznat. (Např. lidské ucho vnímá jen zvuk v rozmezí 20 Hz – 20 kHz přičemž horní hranice se s věkem snižuje.) Tato metoda se označuje jako *percepční kódování*. Podobné principy používá např. JPEG.

2.1.1 MPEG1 audio komprese

Standard MPEG1 neobsahuje přesnou specifikaci enkodéru, ale obsahuje jen principy a algoritmy, kterými lze odstranit nepotřebné části zvuku. Díky tomu existuje mnoho enkodérů lišících se kvalitou výstupních souborů.

Blokové schema enkodéru je na obr. 2.1.



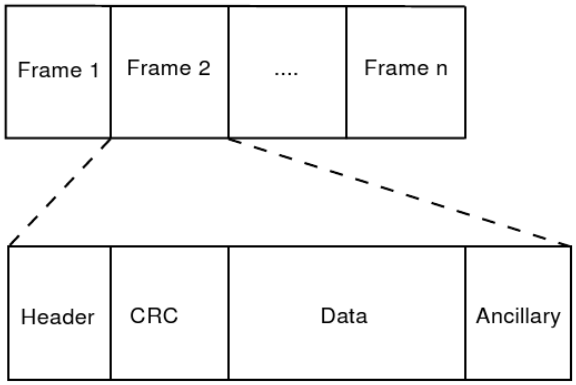
Obrázek 2.1: Blokové schema enkodéru

Nejdříve se spektrum vstupního zvuku rozdělí do několika podpásem. Tyto podpásma korespondují s tzv. kritickými pásmy hlemýždě ve vnitřní části lidského ucha (podrobněji v [21] kap. 2.2). Paralelně s filtrováním enkodér používá psychoakustický algoritmus, který poskytuje optimální poměr signál-maska (signal to mask ratio – SMR). Poté jsou frekvenční

vzorky nakvantovány a zakódovány. Zde se enkodér snaží o minimalizaci kvantizačního šumu a využívá data z psychoakustického modelu pro maskování frekvencí. Nakonec jsou data naformátována do bitového proudu.

2.1.2 Struktura souboru

MPEG Audio soubor se fyzicky dělí na rámce, které obsahují hlavičku (32 bitů), CRC součet (16 bitů), audio data a další doplňková data (viz. obr. 2.2). Velikost každého rámce není konstantní a rámec končí hlavičkou dalšího rámce.



Obrázek 2.2: Struktura MPEG1 audio souboru

Hlavička rámce

Hlavička obsahuje 4 byty.

Velikost [bit]	*Název*	*Popis*
12	Sync header	Synchronizační část, obsahuje 12 jedniček
1	Version	Verze použitého standardu
2	Layer	Verze kódovacího schematu
1	Error protection	Udává, zda jsou přítomny kontrolní součty
4	Bit rate	Rychlost datového proudu 0–448 kbit/s
2	Sample rate	Vzorkovací frekvence
1	Padding	Vyplnění nevyužitých bitů v rámci
1	Private bit	Volné použití
2	Channel mode	Označuje kódování dat do kanálů (mono, joint-stereo…)
2	Mode extension	Použito při joint-stereo, identifikuje joint-stereo metodu
1	Copyright	
1	Original	
2	Emphasis	Vyjadřuje zda jsou zvýrazněny frekvence nad 3,2 kHz

Tabulka 2.1: Hlavička MPEG1 rámce[21]

Podrobnější popis hlavičky lze nalézt v [21] nebo přímo v normě ISO/IEC-11172, která ale není volně přístupná.

Stereo módy

- *Dual channel* – Datový tok je rovnoměrně rozdělen mezi oba kanály, tento mód je vhodný zejména pro vzájemně nezávislé kanály.
- *Stereo* – Od dual channel se liší možností přidělit každému kanálu jiný datový tok. Když je v jednom kanálu ticho, druhému kanálu se přidělí větší datový tok.
- *Joint stereo* – V tomto módu je využita podobnost obou kanálů. Vytvoří se součtový a rozdílový kanál. Dojde ke zvýšení celkové kvality na úkor částečné ztráty sterea. Do datového toku 192 kbit/s je tento mód jediný, který poskytne dostatečnou kvalitu.

2.1.3 Vrstvy

MPEG1 standard je rozdělen na 3 vrstvy (Layer I až Layer III). Od Layer I do Layer III roste složitost a efektivita komprese zvuků, ale klesá rychlost kódování a dekódování. Vrstvy jsou zpětně kompatibilní, takže např. Layer II dekodér umí přehrát Layer I, ale neumí Layer III.

- Layer I je nejjednodušší schéma. Frekvenční spektrum je rozděleno na 32 pásem, přičemž nižší frekvence mají užší pásmo (to ale neodpovídá kritickým pásmům ucha)[5]. Každé pásmo má 12 vzorků, každý rámec tedy obsahuje 384 vzorků.

  Tato vrstva byla výhodná pro telekonference nebo real-time střih, protože jej bylo možné kódovat v reálném čase i v roce cca 1990. S růstem výkonu hardwaru se tato vrstva přestala používat a dnes je již zastaralá.[15]

- Layer II je kompromis mezi kvalitou, rychlostí a kompresním poměrem. Tato vrstva je odvozena od kodeku MUSICAM. Rozšiřuje Layer I definováním 3 měřítek (scale factors) pro každé pásmo[24]. Rámec tedy obsahuje $3 \cdot 12 \cdot 32 = 1152$ vzorků. Technické detaily lze nalézt v [15].

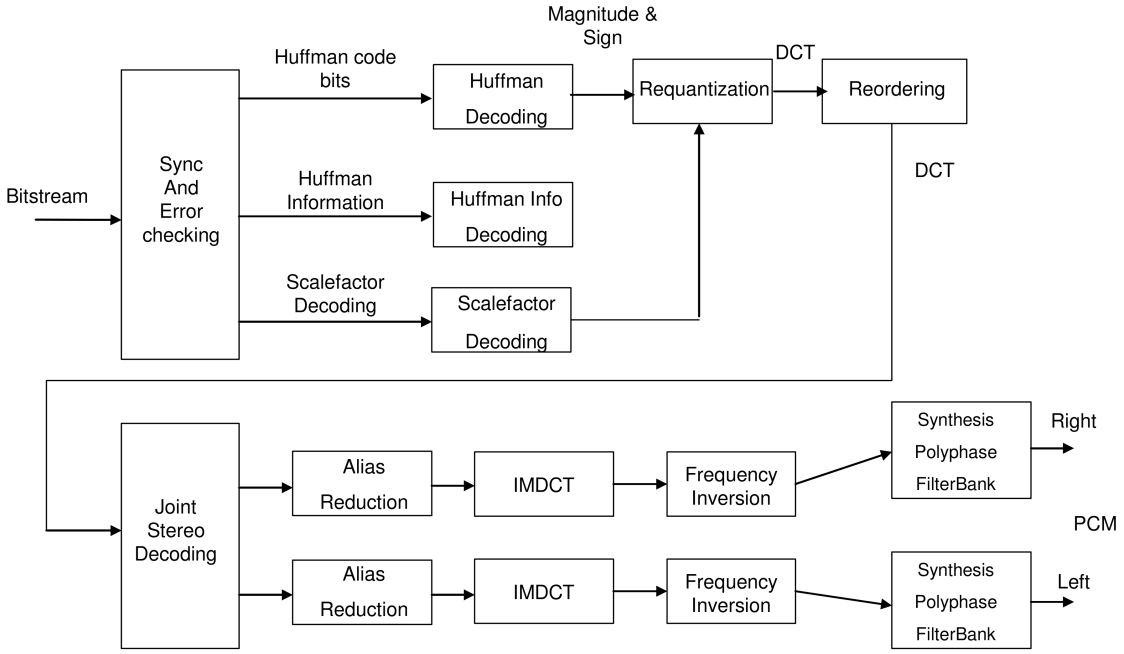
- Layer III (MP3) je od začátku vytvářena tak, aby poskytla přijatelnou kvalitu při datovém toku 64 kbit/s, což je propustnost jednoho ISDN kanálu. Vrstva je odvozena od kodeku ASPEC. Pro zvýšení frekvenčního rozlišení používá modifikovanou diskrétní kosinovou transformaci (MDCT [13]) a další pokročilé techniky viz. [15]. Výsledný bitový proud je na rozdíl od vrstev I a II ještě zakódován Huffmanovým kódem s variabilní délkou, což vede k dalšímu snížení datového toku.

2.1.4 Dekódování MP3[3]

Na obrázku 2.3 je zobrazeno blokové schema MP3 dekodéru. Vstupní data jsou nejdříve rozdělena na jednotlivé rámce a otestována na přítomnost chyb pomocí CRC součtu, který je součástí každého rámce. Následně jsou z postranních informací extrahována „měřítka“ (scale factors), které jsou později potřeba k rekvantizaci vzorků.

Hlavní audio data jsou kódována Huffmanovým kódem. Protože takto zakódovaná data mají různou délku, mohou být data obsažena i v sousedních rámcích. Proto blok Huffmanova dekodéru potřebuje přístup i k předcházejícím rámcům.

Dalším krokem dekódovacího procesu je rekvantizace, při které se dekódované vzorky upraví pomocí měřítek. Následně jsou data zařazena do frekvenčních pásem a takto vstupují do stereo dekodéru, který data rozdělí do dvou kanálů. Data každého kanálu projdou blokem redukujícím alias, který může vzniknout v enkodéru.



Obrázek 2.3: Blokové schema MP3 dekodéru

Další blok provede inverzní kosinovou transformaci (druh zpětné Fourierovy transformace [13]). Nyní zbývá již jen převést vzorky na pulzně kódový (PCM [16]) signál, který lze již snadno převést na analogový signál.

2.2 Formát WMA

Formát WMA je komprimovaný zvukový formát vytvořený firmou Microsoft jako součást Windows Media. Původně měl nahradit formát MP3, ale jeho konkurenceschopnost byla příliš nízká. To se změnilo s vydáním verze WMA9, které je již na úrovni nejvyspělejších kodeků.

Problémem tohoto kodeku je sice příliš velké ořezávání vysokých frekvencí při nižších datových tocích, ale nevytváří při nízkých datových tocích tolik artefaktů jako formát MP3. Proto je pro mluvené slovo lepší použít kodek WMA než kodek MP3.

K tomuto formátu neexistuje otevřená specifikace, což jeho používání omezuje. Pro verzi WMA9 existuje jen jeden enkodér, který je integrovaný ve Windows Media Player. Pro verzi WMA8 a nižší existuje implementace vytvořená pomocí reverzního inženýrství v rámcio projektu FFmpeg.

Postup dekódování tohoto formátu není volně přístupný. Musí se použít originální *msco-dec*.

2.3 Formát OGG Vorbis

Vorbis je svobodný ztrátový zvukový kodek, který by měl nahradit formát MP3. Nejčastěji bývá uložen v kontejneru Ogg a proto s ním také bývá zaměňován.

Formát je velmi dobře popsán v oficiální dokumentaci [1].

Algoritmus enkódování a dekódování je podobný jako u formátu MP3. Používá se modifikovaná diskrétní kosinová transformace [13] pro konverzi z časové domény do frekvenční oblasti. Výsledná data jsou upravena pomocí psychoakustického algoritmu, nakvantována a zakódována entropickým codebook [12] algoritmem. Kvalita Vorbis kodeku je lepší než MP3.

Ukládání metadat je u tohoto kodeku problematické, protože pro ně zatím neexistuje oficiální standard. Používají se Vorbis komentáře (*Vorbis comments*), které mají podobnou strukturu jako ID3v2 tagy v MP3 souborech a jsou kódovány pomocí UTF-8 standardu.

Kapitola 3

Návrh hardware modulu

3.1 Rozšíření zadání

Ihned při první konzultaci s mým vedoucím práce jsem se rozhodl si zadání trochu rozšířit, protože jen přehrávání souborů MP3 se mi jevilo jako příliš snadné. Nejdříve jsem si rozšířil zadání o 2,5" pevný disk, ze kterého by bylo možné načítat MP3 soubory, a dále o LCD displej, který by zobrazoval informace o přehrávané skladbě. Rozšíření o USB rozhraní přišlo až s vývojem druhé verze modulu (viz. dále v sekci 3.5). Oficiální zadání jsem ale rozšiřovat nechtěl, protože nebylo jisté, zda všechna rozšíření stihnu implementovat.

3.2 Výběr komponent

3.2.1 Audio kodek

Obvodů, které umějí dekódovat formát MP3, je velké množství. Snad každá firma vyrábějící integrované obvody má nějaký takovýto dekodér v nabídce. Při výběru konkrétního obvodu jsem se zaměřil na schopnost dekódovat i jiné formáty než jen MP3, protože tento je již dnes překonán a v praxi se začínají používat i jiné formáty (OGG, AAC, WMA9). Dalším kritériem byla jednoduchost komunikace s obvodem (nejlépe přes sériové rozhraní) a možnost provádět přímo v čipu úpravu hlasitosti, basů a výšek, aby nebylo nutné připojovat k dekodéru další obvody.

Po prostudování nabídky dostupných dekodérů jsem zvolil čip s označením VS1053[23] od finské firmy VLSI Solution. Tento dekodér je schopen přehrát širokou škálu formátů, obsahuje jednoduchý ekvalizér a výstupní zesilovač, takže je možné připojit přímo k čipu sluchátka. Rozhodujícím faktorem pro výběr tohoto dekodéru ovšem byla možnost data v průběhu dekódování měnit pomocí vlastního softwaru, který lze nahrát do dekodéru. Díky tomu lze implementovat např. ekvalizér nebo spektrální analýzu, jejíž výsledek je možné zobrazovat během přehrávání na připojeném displeji.

Jeho dalšími výhodami jsou možnost nahrávat zvuk buď z mikrofonu nebo linkového vstupu do formátů OGG nebo ADPCM, malé množství potřebných externích součástek a sériové rozhraní s 2 kB velkou vyrovnávací pamětí.

3.2.2 LCD displej

Při výběru LCD displeje jsem nejprve uvažoval o displeji z mobilního telefonu (Nokia 6100), protože na trhu nebyl vyhovující malý a levný displej s integrovaným řadičem. Přibližně po

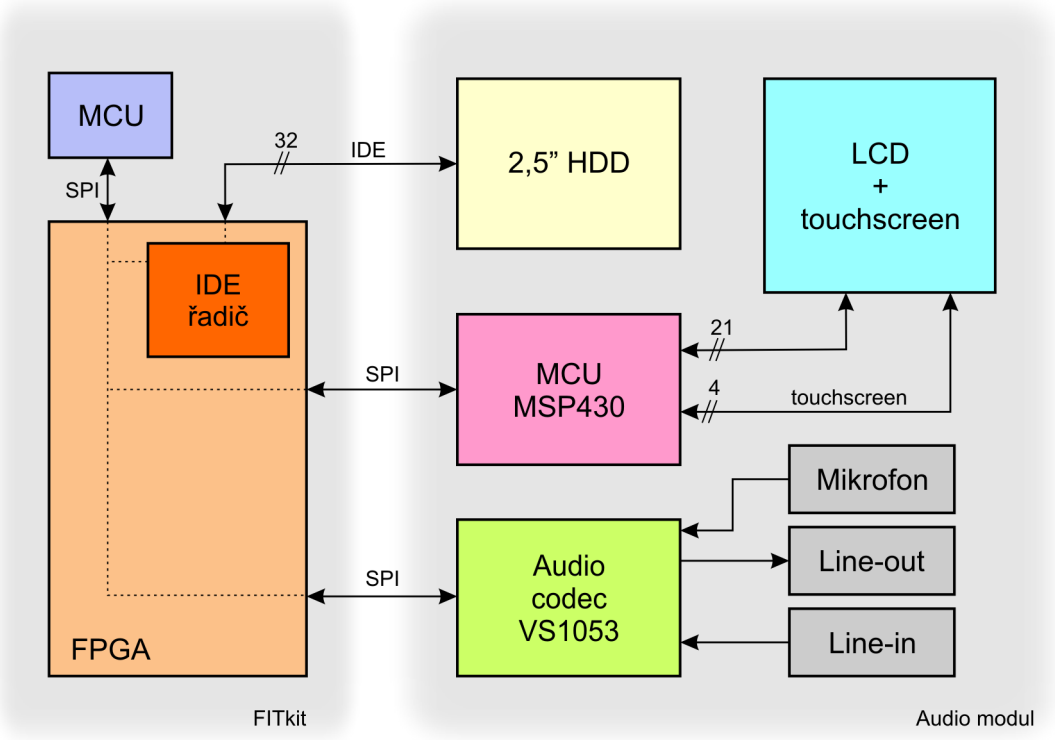
měsíci jsem ale u firmy Farnell našel novinku – dotykový LCD displej s úhlopříčkou 2,8" a rozlišením 320 × 240 pixelů za rozumnou cenu 60 €.

Jedná se o LCD modul C0283QGLH-T[4] od firmy DENSITRON. Displej je typu OLED s aktivní maticí (spínání každého pixelu je prováděno vlastním tranzistorem) a je schopen zobrazit až 2^{18} barev. LCD modul obsahuje řadič[17] s pamětí, který přímo ovládá matici LCD displeje, a je možné k němu přistupovat přes paralelní i sériové rozhraní.

Displej je vybaven odporovou dotykovou vrstvou, která usnadní ovládání a zvýší uživatelský komfort.

3.3 První verze modulu – popis zapojení

Blokové schéma je na obrázku 3.1. Elektrotechnické schéma se nachází v příloze A.



Obrázek 3.1: Blokové schéma první verze audio modulu

3.3.1 Výběr rozhraní řadiče displeje

Řadič displeje je možné ovládat pomocí až tří rozhraní – paralelního, sériového a RGB.

Při použití RGB rozhraní musejí být data stále obnovována (nepoužívá se paměť v řadiči), přičemž se využívá řádkové a snímkové synchronizace. Použití tohoto rozhraní je pro naše účely zcela nevhodné, je vhodné např. k připojení kamery nebo jiného zařízení, které disponuje tímto výstupem.

Nejdříve jsem zamýšlel využít sériového SPI rozhraní a připojit jím displej k FPGA na FITkitu. Displej by tak byl ovládán mikrokontrolérem na FITkitu a vykreslování některých

grafických objektů by mohlo být akcelerováno v FPGA. Tuto možnost jsem ale po zjištění maximální řadičem podporované rychlosti sériového rozhraní (perioda hodinového signálu minimálně 130 ns) zamítl, protože by tak i při 100 % vytížení sběrnice bylo možné překreslit displej jen přibližně 2× za vteřinu, v případě použití jen 16-ti bitových barev 4× za vteřinu. Tyto hodnoty by se v praxi ještě zhoršily, protože při každém zápisu do paměti je nutné obsluhovat pin CSB a přitom musí být hodinový signál v klidu.

Řešením se zdálo být použití paralelního rozhraní, které by rychlost přenosu podstatně zvýšilo. Tomu ovšem bránil nedostatek vývodů na FITkitu, protože jejich podstatnou část zabralo rozhraní pro připojení pevného disku. Jediným východiskem z této situace bylo buď nepřipojení pevného disku nebo použití dalšího mikrokontroléru na rozšiřujícím modulu. Vzhledem k tomu, že mikrokontrolér na FITkitu není příliš výkonný a nejspíš by měl problémy posílat data do audio kodeku a zároveň obsluhovat displej, jsem se rozhodl pro použití dalšího mikrokontroléru, který by obsluhoval pouze LCD displej a dostával příkazy z FITkitu přes sériové rozhraní.

3.3.2 Mikrokontrolér pro obsluhu displeje

Vybíral jsem mikrokontrolér ze stejné řady jako je na FITkitu, aby se nijak nelišil způsob programování od programování mikrokontroléru FITkitu. Vybral jsem oproti FITkitu výkonnější verzi, která může běžet až na frekvenci 16 MHz, konkrétně typ MSP430F248, který má navíc ještě dvakrát větší operační paměť.

Připojení displeje k MCU

Řadič displeje se připojuje k mikrokontroléru 5 řídícími vodiči: *chip select*, *register select*, *reset*, *read/write* a *enable strobe*. Poslední dva lze přepnout i do módu, ve kterém jsou dva signály *enable strobe*, jeden pro čtení a druhý pro zápis.

Data lze přenášet po 8, 16 nebo 18-ti vodičích. Při použití méně datových vodičů než jakou bitovou hloubku barev chceme zobrazovat se data posílají ve více fázích. Barevnou hloubku jsem zvolil na 16 bitů, protože rozdíl v kvalitě obrazu při použití 16-ti a 18-ti bitových barev nebude příliš velký a také proto, že mikrokontrolér je 16-ti bitový, a s 18-ti bity na pixel by se obtížně pracovalo. Data jsou tedy přenášena po 16-ti vodičích, přičemž pro zelenou barvu, na kterou je lidské oko nejcitlivější, je vyhrazeno 6 bitů, pro modrou a červenou barvu potom po 5 bitech.

Programovací rozhraní

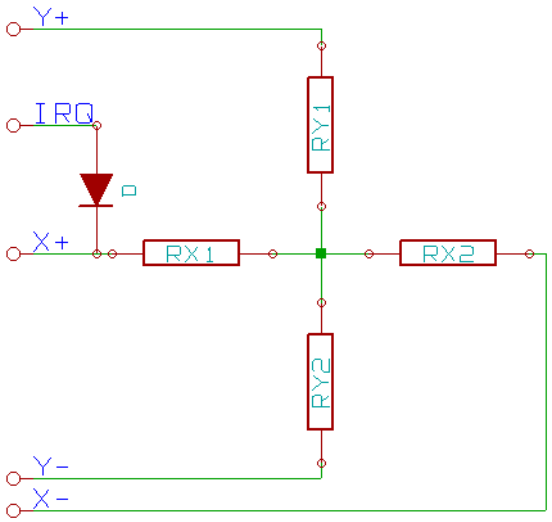
Vzhledem k tomu, že jsem na audio modulu použil mikrokontrolér ze stejné rodiny jako je na FITkitu, vytvořil jsem i stejný programovací JTAG konektor jako se nachází na FITkitu, abych mohl použít již osvědčené nástroje.

3.3.3 Generátor záporného napětí

OLED displej potřebuje ke své funkci připojení napětí +4,6 V a -4,4 V. K tvorbě tohoto napětí jsou výrobcem LCD displeje doporučeny speciální obvody. Všechny se ovšem vyrábějí pouze v QFN pouzdru, což si vyžádalo osazení desky ve specializované externí firmě. Použil jsem generátor TPS65136[20] od firmy Texas Instruments, který byl jako jediný z doporučovaných dostupný v kusovém množství. Jeho zapojení jsem převzal z katalogového listu.

3.3.4 Touch screen

Dotykový displej obsahuje 2 tenké odporové vrstvy oddělené malou mezerou. Když se dotkneme nějakým předmětem displeje, vrstvy se vodivě propojí a vytvoří tak odporový dělič (obr. 3.2). Odporový touch screen se připojuje 4 vodiči k A/D převodníkům mikrokontroléru. Postup získání souřadnic dotyku je popsán v kapitole 5.4.2.



Obrázek 3.2: Touch screen

Generování přerušení při dotyku na dislej

V klidovém stavu je na jednu vrstvu přivedeno napětí, na druhou vrstvu je připojena dioda D. Když dojde k propojení vrstev, katoda diody D se uzemní a vznikne tak sestupná hrana, která může sloužit ke generování žádosti o přerušení.

3.3.5 Audio kodek

Zapojení audio kodeku vychází z jeho katalogového zapojení. Přidal jsem jen možnost přepínání mezi mikrofonním a linkovým vstupem pomocí jumper přepínače. Dále jsem vyvedl GPIO piny a přidal tak možnost připojit k audio kodeku externí zařízení. Pokud tyto piny nebudou použity, musí se propojkami uzemnit.

Frekvenční filtry

Ke zvukovému výstupu jsou připojeny dolnofrekvenční propusti tvořené RC články, které odfiltrují frekvence od 800 kHz výše. Takto vysoké frekvence mohou vznikat v D/A převodníku kodeku.

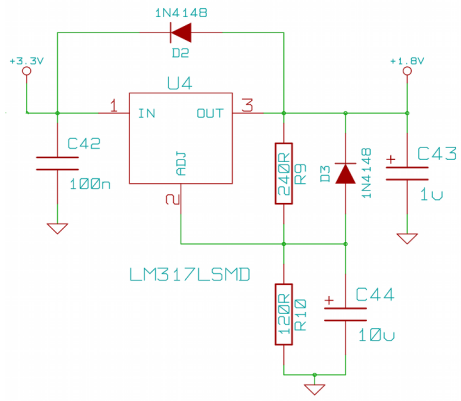
Zvukový vstup je filtrován horní i dolní propustí a propouští frekvence v rozmezí od 300 Hz do 30 kHz.

Rozhraní

Rohraní audio kodeku se skládá ze standardního čtyřvodičového SPI rozhraní, vodiče DCS, který určuje zda jsou přes SPI rozhraní posílána data nebo instrukce, a vodiče DREQ. Při posílání dat je jejich proud před zpracováním ukládán v kruhové vyrovnávací paměti (FIFO frontě) o velikosti 2048 bytů. Jakmile se v této frontě uvolní 32 a více bytů změní se napěťová úroveň vodiče DREQ a řídící mikrokontrolér je tak informován, že může poslat další data.

Napájení jádra

Jádro kodeku vyžaduje napájecí napětí 1,8 V. Toto napětí se však na FITkitu nevyskytuje, takže jej je třeba vytvořit. K tomu jsem použil stabilizátor LM317L[6]. Jeho zapojení jsem rezistory R_9 a R_{10} upravil tak, aby výsledné stabilizované napětí bylo 1,8 V (obr. 3.3). Diody D_2 a D_3 slouží jako ochrana stabilizátoru při vypnutí napájení.



Obrázek 3.3: Stabilizátor napětí 1,8 V.

Stabilizátor udržuje mezi vývody *Adjust* a *Output* napětí 1,2 V. Stejné napětí je tedy i na rezistoru R_9. Hodnotu tohoto rezistoru zvolíme tak, aby jeho odpor společně s odporem rezistoru R_{10} tvořili zátěž produkující alespoň minimální proud, který je potřebný pro správnou funkci stabilizátoru (10 mA). Hodnotu rezistoru R_{10} vypočítáme podle následujícího vzorce (proud z vývodu *Adjust* 50 μA můžeme zanedbat):

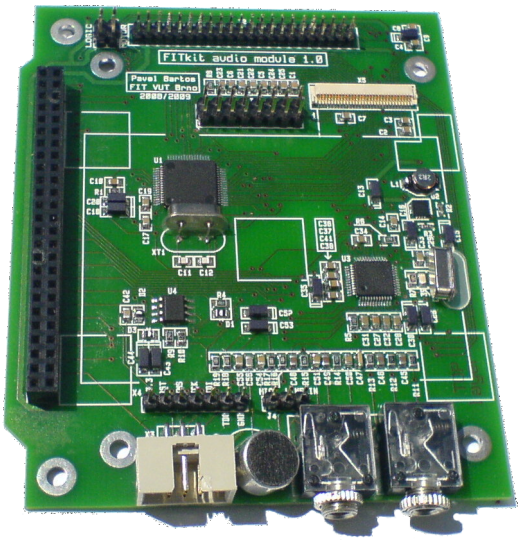
$$R_{10} = (\frac{U_{out}}{1,2} - 1) \cdot R_9$$

3.4 Deska plošných spojů (první verze modulu)

K návrhu desky plošných spojů jsem použil opensource balík aplikací KiCAD. Deska plošných spojů je oboustranná s prokovenými otvory a má přibližně velikost 2.5" pevného disku, který je umístěn z její spodní strany.

Modul je vytvořen tak, aby ho bylo možné snadno a pevně přichytit k FITkitu. K tomu slouží 50-ti pinový konektor X1 osazený ze spodní strany desky, který se připojí na konektor JP10 na FITkitu. Spojení je jištěno 2 distančními sloupky v rozích desky.

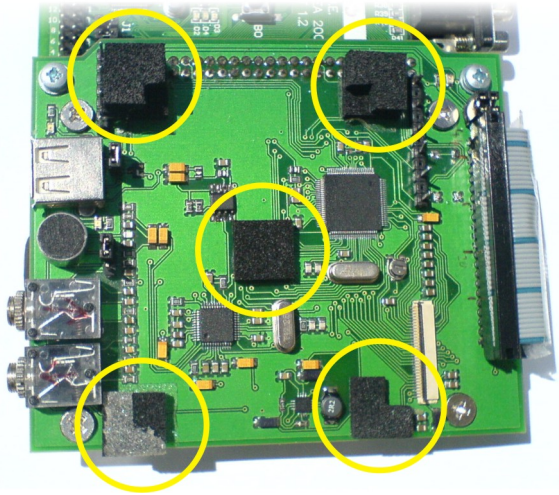
Všechny ostatní součástky jsou umístěny na vrchní straně desky, protože na spodní straně by překážely pevnému disku. Ten je připevněn 4 šrouby a pomocí několika kusů



Obrázek 3.4: Osazená první verze desky plošných spojů

pěnových podložek je vymezena přibližně 3 mm velká mezera mezi deskou a diskem, která slouží k chlazení elektroniky disku. Disk se připojuje velmi krátkým plochým kabelem ke konektoru X2 na okraji desky. U tohoto konektoru jsou i dvě propojky, kterými je možné ovládat napájení disku.

LCD displej je umístěn uprostřed desky. Protože ho není možné připevnit k desce napevno, je přichycen v každém rohu několika samolepícími pěnovými podložkami (obr. 3.5). Displej je tak mírně vyvýšen nad desku a pod ním tak mohou být umístěny další součástky.



Obrázek 3.5: Uchycení LCD displeje

Konektor LCD displeje není příliš běžný a tak jsem při návrhu desky musel jeho pouzdro ručně nadefinovat. V jeho blízkosti jsou umístěny kondenzátory, které jsou vyžadovány řadičem displeje, a také generátor záporného napětí.

Při rozmísťování součástek audio části jsem se snažil navrhnout plošné spoje tak, aby

se nekřížily s napájecími a nedocházelo tak k rušení. To se nepodařilo zcela, protože audio konektory jsou umístěny až na okraji desky a napájecí spoje nebylo možné vést jinou cestou. K rušení ale nedochází.

K oživování této verze modulu jsem se však nedostal, protože ve firmě, kde desku vyráběli a osazovali, osadili špatně konektor X1 a při pokusu o jeho odpájení se odtrhly některé plošné spoje, takže deska byla dále nepoužitelná.

3.5 Druhá verze modulu

V době, kdy byla první verze ve výrobě, jsem objevil na internetu grafickou knihovnu[9] pro mikrokontroléry PIC, která je zdarma ke stažení. Její použití je ovšem licencí povoleno jen pro mikrokontroléry PIC. Plánoval jsem se touto knihovnou inspirovat ke tvorbě vlastní knihovny pro MSP430, která by ale musela být jednodušší, protože procesory MSP430 mají nižší výkon.

Nicméně po zničení první desky jsem se po poradě se svým vedoucím rozhodl vytvořit druhou verzi modulu, ve které použiji místo mikrokontroléru MSP430 mikrokontrolér PIC32 a budu tak moci použít grafickou knihovnu[9] od firmy Microchip.

3.5.1 Mikrokontrolér PIC32

Použil jsem konkrétně typ PIC32MX440F128L[10]. Jak označení napovídá jedná se o 32-bitový mikrokontrolér a disponuje 128 kB flash paměti. Může běžet až na frekvenci 80 MHz, která je více než dostatečná a umožňuje využít potenciál grafické knihovny naplno.

Mezi jeho rozhraní patří i USB, což jsem samozřejmě využil a je tak možné načítat zvukové soubory z USB flash paměti, a také speciální 16-bitový I/O port, který lze nastavit tak, že automaticky generuje signály *read/write* a *enable strobe*. Je tak možné poslat data řadiči displeje jen jedním zápisem do registru.

Disponuje také 4 kanálovým DMA řadičem, což umožní přehrát i zvukové soubory s velkým bitovým tokem (více v kapitole 5.1.5).

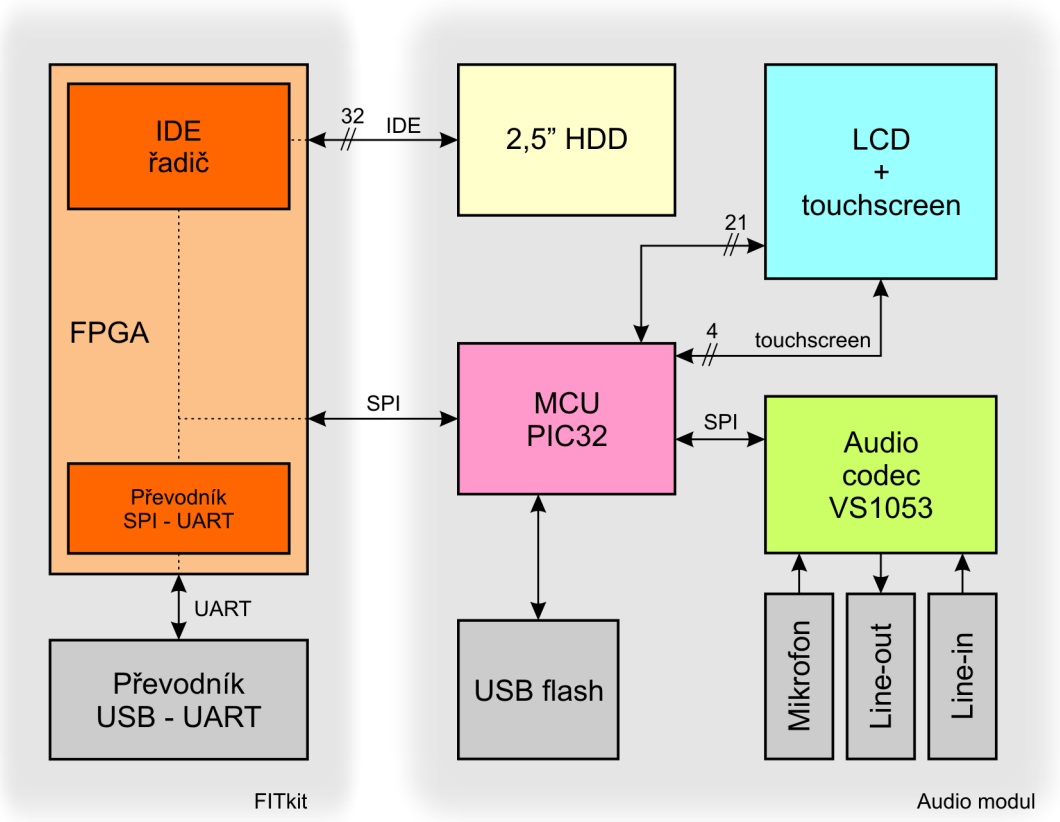
3.5.2 Rozdíly proti první verzi

Vysoký výkon mikrokontroléru PIC32 mi umožnil nepoužívat procesor na FITkitu a celou vzájemnou komunikaci komponent na modulu tak zjednodušit. Procesor na FITkitu slouží jen k nahrání konfigurace do FPGA, kde je umístěn IDE řadič a převodník SPI rozhraní na asynchronní sériové rozhraní UART, které lze připojit přes USB převodník k PC a slouží jen k testovacím výpisům. Pokud se tedy spokojíme s načítáním dat jen z USB flash, je audio modul po přivedení napájení zcela samostatný a nezávislý na FITkitu.

Připojení řadiče LCD displeje je podobné jako u první verze (viz. 3.3.2). Paralelní sběrnice je ale připojena k 16-ti bitovému portu, který také zároveň automaticky ovládá i signály *read/write* a *enable strobe*. V kódu programu tedy stačí jen ovládat signály *chip select* a *register select*, což komunikaci zrychlí a zjednoduší.

Audio kodek je připojen stejně jako u první verze přes SPI rozhraní.

Zapojení USB rozhraní je s mikrokontrolérem PIC32 jednoduché, protože datové piny z mikrokontroléru lze připojit přímo ke konektoru. Připojení napájecího napětí 5 V lze ovládat buď přímo pomocí jumper přepínače nebo přes mikrokontrolér vývodem označeným *USB_ON*, který pomocí MOSFET tranzistoru připojuje a odpojuje napájení USB.



Obrázek 3.6: Blokové schéma druhé verze audio modulu

Programovací rozhraní jsem vytvořil pro oba možné způsoby programování (JTAG a ICSP), neboť jsem v době návrhu nevěděl, jaký budu mít k dispozici programátor. Ze stejného důvodu jsou u rozhraní ICSP vyvedeny i dvě úrovně napětí 3,3 V a 5 V.

Protože po přidání USB konektoru a programovacích rozhraní nebyl na desce dostatek místa, zrušil jsem oproti minulé verzi konektor s GPIO piny audio kodeku, pro které stejně nebylo využití. Tyto piny jsou jen uzemněny přes 100 kΩ rezistor.

3.5.3 Deska plošných spojů

Návrh druhé verze desky vychází z verze první. Ke změnám došlo jen v oblasti mikrokontroléru, kterou jsem musel celou přepracovat.

Při tom jsem se ale dopustil chyby, kdy jsem pouzdro mikrokontroléru typu TQFP100 vybral z knihovny dodávané se softwarem KiCAD. Předpokládal jsem, že pouzdro s takovým označením je jen jedno a nezkontroloval jsem si ho s katalogovým listem výrobce mikrokontroléru, kde jsou uvedeny jeho správné rozměry. Na tuto moji chybu se bohužel přišlo až při osazování, kdy nebylo možné mikrokontrolér osadit. Na svoje náklady jsem tedy nechal vyrobit novou desku.

3.6 Oživení

Po prvním připojení napájecího napětí se rozsvítil displej a žádná součástka nejevila žádné teplotní ani pachové abnormality. První zapojení tedy proběhlo v pořádku.

Při pokusu o připojení programátoru k mikrokontroléru jej ale programátor nerozpozná. Zjišťuji, že je mikrokontrolér trvale resetován. Nejdříve chybně detekuji průraz kondenzátoru RC členu na resetujícím pinu a následně zjišťuji, že jsem špatně vytvořil pouzdro pro resetovací tlačítko, které nyní trvale propojuje resetovací pin se zemí. Po odpájení tlačítka je resetování již v pořádku.

Programátor teď rozpozná typ mikrokontroléru a komunikace přes programovací rozhraní je tedy funkční. Nicméně občas dojde k chybě při programování a programátor hlásí, že není připojeno žádné napětí. Pomocí multimetru zjišťuji, že na pinu, kde má být napětí 3,3 V je jen asi 1,5 V. Ovšem když na pin trochu zatlačím změní se napětí na 2,5 V. Usuzuji, že jsou u konektoru špatně prokovené otvory, protože konektor je zapájen jen ze spodní strany, ale spoje jsou připojeny na vrchní straně (vzhledem k umístění konektoru je nebylo možné připojit na spodní straně). Po prohřání pinu, aby pájka zatekla více do otvoru, se podařilo opravit 2 spoje, které se používají k programování. U dalšího se mi podařilo za použití mírného násilí zapájet jej i z vrchní strany. Další piny s napájením se opravit nepodařilo, ale naštěstí jsou piny s napájecím napětím vyvedeny na FITkitu a tak mohu napájení programátoru připojit k němu.

Další problémy s oživením komponent jsem již řešit nemusel.

Kapitola 4

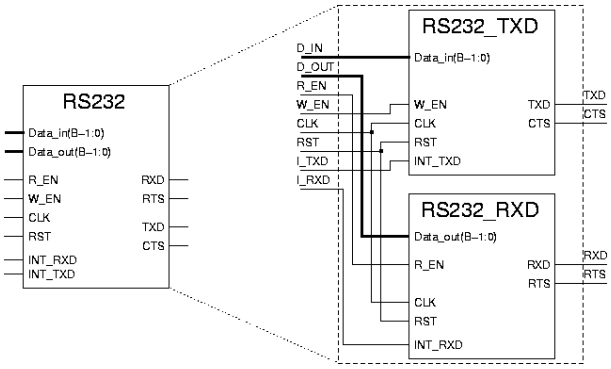
Konfigurace FPGA

4.1 Propojení modulu a FITkitu

K propojení mikrokontroléru a FPGA na FITkitu jsem použil stejné rozhraní i komunikační protokol[22], který se používá mezi mikrokontrolérem na FITkitu a FPGA. Mělo by tedy být možné ovládat modul i z mikrokontroléru FITkitu.

4.2 Konzolový výstup

Absence debuggeru a nutnost mít alespoň nějaký testovací výstup mě donutila vytvořit v FPGA převodník SPI rozhraní na asynchronní sériové RS-232 rozhraní. To je přes USB převodník připojeno k PC, kde na terminálu mohu přijímat ladící výpisy odeslané mikrokontrolérem.



Obrázek 4.1: Převodník SPI - UART

Pro převodník jsem použil komponentu `serial`[7] (obr. 4.1) z knihovny FITkitu. Data jsou posílána rychlostí 921600 baudů a jsou zabezpečena lichou paritou.

4.3 IDE řadič

Použil jsem řadič od mého spolužáka Bc. Stanislava Sigmunda, který jej vytvořil v roce 2007 v rámci své bakalářské práce[18]. Zapojení jeho vývodů jsem ale musel upravit, protože k připojení disku nemám k dispozici všechny piny konektoru X na FITkitu. Vývody, které

jsou trvale ve stavu vysoké impedance nebo které jsou trvale uzemněny jsem na konektor nepřipojoval. Jejich případné uzemnění jsem realizoval až na desce plošných spojů modulu.

Kapitola 5

Vývoj softwaru pro mikrokontrolér na modulu

Firma Microchip (výrobce mikrokontroléru) doporučuje vývojářům používat jejich integrované vývojové prostředí MPLab s kompilátorem C32. Protože používám operační systém Linux, je pro mě velkou nevýhodou podpora pouze operačního systému Microsoft Windows, v unixových systémech nelze vývojové prostředí spustit ani s pomocí emulátoru Wine. Jediné rozumné řešení tedy bylo nainstalovat Microsoft Windows do virtuálního stroje a celý software diplomové práce vyvíjet v tomto virtuálním stroji.

V průběhu vývoje jsem několikrát narazil na chybné přeložení kódu překladačem. Například po přidání komentáře do zdrojového kódu byl přeložený kód nefunkční. Bohužel se mi ale nepodařilo problém blíže identifikovat, abych ho mohl nahlásit vývojářům.

K programování mikrokontroléru mi byl školou zapůjčen univerzální programátor ASIX Presto. Ten jako jediný vývojový prostředek byl naprosto bezproblémový.

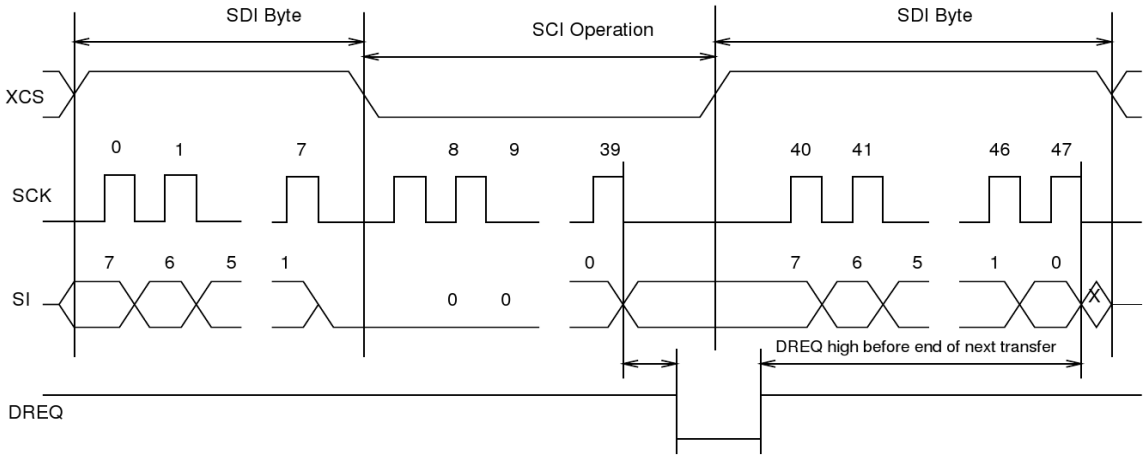
5.1 Zvukový výstup

5.1.1 Komunikace s audio kodekem

S kodekem lze komunikovat dvěma rozhraními, z nichž jedno je zavržené (deprecated). Obě jsou sériová, ale staré zavržené používá oddělené vodiče pro instrukce a pro data. Použil jsem tedy rozhraní nové, které je i jednodušší, ale celou dobu vývoje jsem se potýkal s tím, že téměř všechny příklady v dokumentaci kodeku používají staré zavržené rozhraní.

Nové rozhraní používá pro data i instrukce stejné sériové rozhraní, ale pomocí signálů *chip select* a *data chip select* lze rozlišit instrukce a data. Dále je možné ještě uspořit jeden vodič nastavením kodeku do módu, kdy si signál *data chip select* vytváří sám vnitřně invertováním signálu *chip select*.

Na obrázku 5.1 je schéma komunikace s kodekem při posílání 2 bytů dat proložených 1 instrukcí. Je vidět, že se způsob posílání dat a instrukcí liší jen počtem bitů a v úrovni signálu *chip select*. Dále je vidět, že data jsou měněna při sestupné hraně a vzorkována při vzestupné hraně. Není problém nastavit sériové rozhraní mikrokontroléru do takovéhoto módu, ale už není možné nastavit klidový stav hodinového signálu na nízkou úroveň, což je dle schématu komunikace vyžadováno. Vždy mezi jednotlivými přenosy dojde k přechodu hodinového signálu na vysokou úroveň, což kodek interpretuje jako přenos dalšího bitu. Z tohoto důvodu je potřeba mít oddělené signály *chip select* a *data chip select* a vždy po odeslání instrukce musí oba signály přejít do vysoké úrovně a resetovat tak sběrnici.

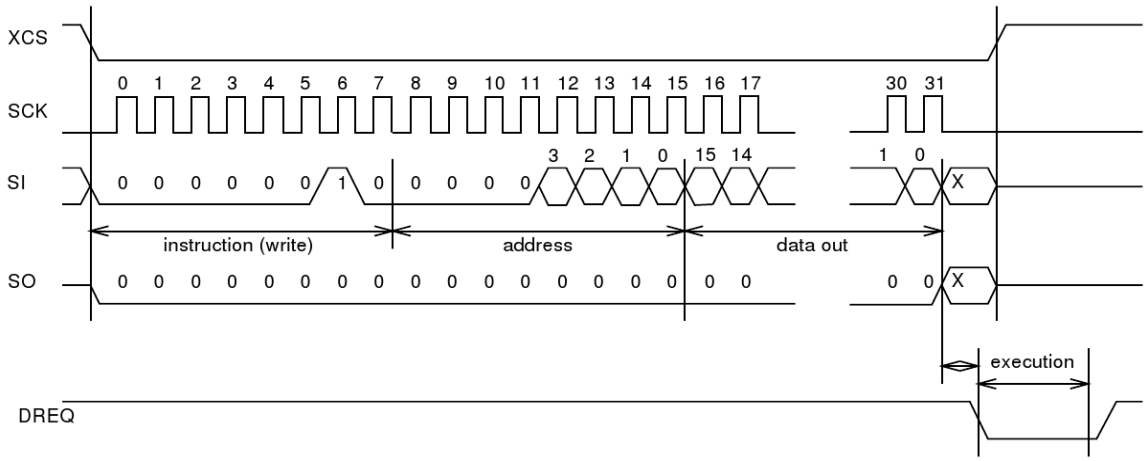


Obrázek 5.1: Schéma komunikace

Sériové rozhraní audio kodeku jsem připojil k prvnímu SPI rozhraní mikrokontroléru. Jeho rychlost jsem nastavil na $\frac{1}{8}$ rychlosti vnitřní sběrnice.

5.1.2 Základní funkce pro komunikaci s kodekem

Zápis dat do registru



Obrázek 5.2: Zápis do registru

Zapsání 16-ti bitů dat do registru kodeku vyžaduje odeslání celkem 32-bitů. V prvním bytu specifikujeme číslem 2, že chceme do registru zapisovat, druhý byte je adresa registru, kam chceme data zapsat. Třetí a čtvrtý byte obsahuje data k zapsání v pořadí big-endian.

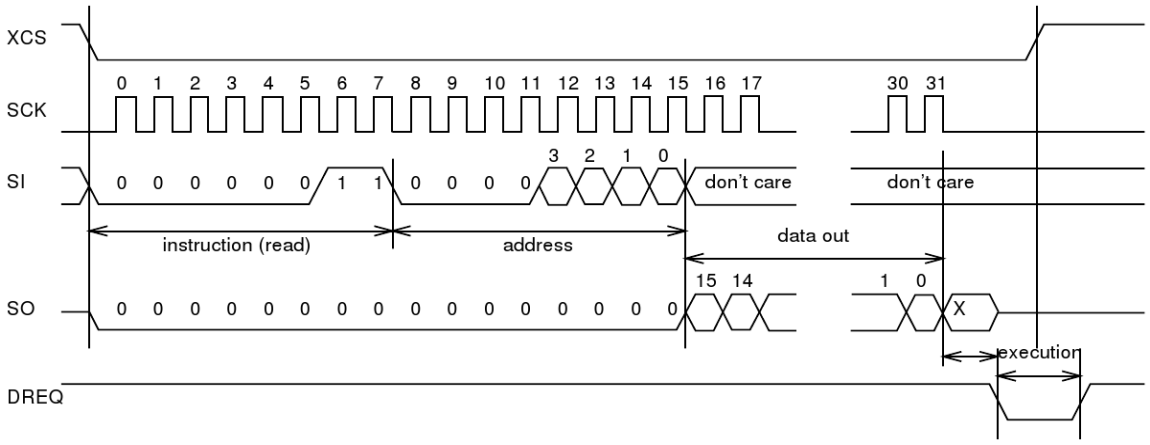
Ukázka zdrojového kódu:

```
while(A_DREQ == 0) {}                    //čekáme, až bude kodek připraven
while(SPI1STATbits.SPIBUSY) {}           //čekáme na minulý přenos
A_CS = 0;                                //budeme přenášet instrukce
```

```
WriteSPI1(((0x0200 | reg) << 16) | data); //začne odesílat data
while(SPI1STATbits.SPIBUSY) {}           //čekáme na dokončení přenosu
A_CS = 1;                                //nic se nepřenáší
```

Před započetím odesílání je třeba vyčkat až nebude aktivní signál *DREQ*, který může signalizovat zaneprázdnění kodeku prováděním předchozí instrukce. Potom se aktivuje signál *chip select*, kterým informujeme kodek, že se budou přenášet instrukce, a následně naplnit buffer sériového rozhraní daty. Vyčkáme odeslání všech 32 bitů a deaktivujeme signál *chip select*.

Čtení dat z registru



Obrázek 5.3: Čtení registru

Princip je stejný jako při zápisu do registru. Jen v prvním bytu se místo čísla 2 zapíše číslo 3. Načtená data z registru nalezneme v posledních 2 bytech příjmového bufferu sériového rozhraní.

Testování funkčnosti kodeku

K otestování funkčnosti kodeku jsem vytvořil funkci `void a_sine_test()`, která odešle jako data sekvenci 0x53EF6E3000000000. Tato sekvence způsobí, že kodek začne generovat funkci sinus o frekvenci 6 kHz a pokud je vše v pořádku, bude v připojených sluchátkách tento signál slyšet. Frekvenci lze měnit hodnotou 4. bytu sekvence podle návodu v dokumentaci[23].

5.1.3 Uživatelský kód v kodeku

Do kodeku lze nahrát uživatelský kód, který má přístup k přehrávaným datům v průběhu dekódování a může je i měnit. Již hotové programy lze nalézt na internetových stránkách výrobce kodeku nebo je možné si naprogramovat vlastní.

Rozhodl jsem se použít spektrální analyzátor ze stránek výrobce. Kód určený k nahrání do kodeku je komprimován metodou RLE, aby zabíral méně paměti v mikrokontroléru. Kód se do kodeku nahrává do paměti RAM. K tomu slouží registr WRAMADDR, který nastavuje adresu do RAM paměti, a registr WRAM do kterého se data zapisují. Po každém zápisu

je registr WRAMADDR inkrementován. Po nahrání je ještě potřeba specifikovat počet a frekvence pásem zápisem těchto hodnot na adresu, která je uvedena v dokumentaci[19] spektrálního analyzátoru. Používám 22 pásem rovnoměrně (v logaritmickém měřítku) rozložených ve spektru slyšitelného zvuku. Jednotlivé frekvence pásem jsou: 30, 41, 55, 75, 102, 139, 188, 255, 347, 470, 639, 867, 1177, 1597, 2168, 2943, 3996, 5424, 7363, 9994, 13567 a 18417 Hz.

Výsledky spektrální analýzy lze načítat z paměti RAM kodeku, kde se na pevně stanovené (viz. dokumentace[19]) adrese nachází pole 16-ti bitových čísel. Každé číslo náleží jednomu pásmu. Bity 5–0 obsahují aktuální výkon v daném pásmu, bity 11–6 špičkový výkon. I když jsou hodnoty uloženy na 6-ti bitech obsahují jen 5-ti bitová čísla.

5.1.4 Nastavení hlasitosti a ekvalizéru

Hlasitost se v kodeku nastavuje zápisem do k tomu určeného registru. Nastavuje se pro každý z kanálů zvlášť, takže pokud nastavíme vyvážení na některý z kanálů, znamená to snížení hlasitosti druhého kanálu. K výpočtu hlasitosti pro jednotlivé kanály se použijí následující příkazy:

```
VolumeLeft  = VolumeMaster - (Balance <= 0)? 0 :  Balance*VolumeMaster/100;
VolumeRight = VolumeMaster - (Balance >= 0)? 0 : -Balance*VolumeMaster/100;
```

Kodek obsahuje i jednoduchý ekvalizér (složitější s více pásmy lze do něj nahrát jako plugin). Lze nastavit zvýraznění basů a utlumení nebo zvýraznění výšek. U obou pásem lze také nastavit hranice, od které se nastavení uplatní. Pro basy jsem vybral 120 Hz a pro výšky 8 kHz. Tyto hranice jsou nastavitelné i za běhu, ale v programu jsem je nastavil napevno, protože uživatelské rozhraní by jinak bylo příliš složité.

5.1.5 Přehrávání souboru

K přehrání zvuku kodekem stačí jen odeslat soubor na jeho datové rozhraní. Kodek má na svém vstupu kruhovou vyrovnávací paměť o velikosti 2 kB. Pokud je v ní alepoň 32 B volného místa, aktivuje se signál *DREQ*, což signalizuje, že můžeme odeslat kodeku 32 B dat.

Vyrovnávací paměť

Protože není možné vždy, kdy si kodek požádá o další data, přistupovat k médiu (může například zrovna probíhat načítání ID3 tagů jiného souboru), vytvořil jsem v mikrokontroléru vyrovnávací paměť, kterou tvoří FIFO fronta, obsahující 300 bloků o velikosti 32 B. Velikost bloku je 32 B proto, aby bylo možné odeslat celý blok, aniž by bylo nutné kontrolovat signál *DREQ*. Počet bloků ve frontě jsem zvolil tak, aby s velkou rezervou nedocházelo k vyprázdnění bufferu a tím k přerušení přehrávání.

Nejdříve byla vyrovnávací paměť naplňována v hlavní smyčce a vyprazdňována v pravidelných časových intervalech daných časovačem. Toto řešení ale bylo funkční jen pro hudební soubory s bitovým tokem přibližně do 160 kb/s. Při vyšších tocích nebyla vyrovnávací paměť dostatečně rychle naplňována a přehrávaný zvuk byl přerušovaný.

DMA přenos

Jediným řešením jak zrychlit přenos dat z paměťového média do kodeku bylo využít DMA řadič mikrokontroléru. Načítání dat z média tím sice urychlit nelze, ale odeslání dat z vyrovnávací paměti na sériové rozhraní se může provádět nezávisle.

DMA přenos ušetří značné množství procesorového času, protože bez něj by se mezi jednotlivými byty muselo čekat na dokončení přenosu. S využitím DMA je možné přehrávat sobory s bitovým tokem 320 kb/s a zřejmě i vyšším, což jsem ale nezkoušel.

Celý přenos dat z média do kodeku je řízen dvěma časovači. Tím je zajištěna nezávislost na hlavním programu, kde může probíhat třeba vykreslování složité komponenty a bez použití časovačů by se nestíhalo doplňovat data do vyrovnávací paměti.

Jedním časovačem se přibližně asi 1500× za vteřinu kontroluje zaplněnost vyrovnávací paměti a případně se do ní načtou data z média. Před přístupem k médiu se ale musí zkontrolovat, zda se k němu právě nepřistupuje v hlavním programu, protože toto načítání zvukových dat je vyvoláno přerušením časovače. K tomuto účelu jsem vytvořil boolean proměnnou, která funguje jako jednoduchý zámek média. V hlavním programu je tato proměnná před přístupem k médiu nastavena a po přístupu zase vynulována. V nastavení této proměnné se v přerušení k médiu nepřistupuje a vyčká se až je toto přerušení znova časovačem vyvoláno, kdy již bude možná médium odemčeno.

Při obsluze přerušení druhého časovače se kontroluje signál *DREQ*, který oznamuje, že kodek vyžaduje poslat další data. Pokud kodek data vyžaduje a neprobíhá zrovna předchozí DMA přenos je DMA řadiči předána adresa odkud má data načítat a adresa sériového rozhraní, kam je připojen audio kodek, a DMA je přenos je spuštěn.

5.1.6 Záznam zvuku

Záznam zvuku se bohužel nepodařilo zprovoznit. V kodeku je totiž chyba a k nahrávání do IMA ADPCM formátu se musí do kodeku dle výrobce nejdříve nahrát opravný kód, který by toto nahrávání měl umožnit. Tak jsem učinil, ale kodek místo zvukových dat odesílá nesmyslná data, která není možné přehrát.

Pomocí kodeku lze nahrávat zvuk i do formátu Ogg Vorbis. K tomu je ale potřeba do kodeku nahrát plugin, který nahrávaná data komprimuje. Aby bylo možné plugin do kodeku nahrát, je třeba ho mít uložený v paměti mikrokontroléru. Tento plugin má ale velikost přibližně 12 kB a tolik místa v mikrokontroléru není k dispozici.

Pokud by se mikrokontrolér vyměnil za typ s větší pamětí, nahrávání by pravděpodobně mělo fungovat. Software mikrokontroléru i uživatelské rozhraní jsou na tuto možnost připraveny.

5.2 USB rozhraní

Ke komunikaci s USB zařízeními je firmou Microchip dodávána knihovna. Je možné, aby mikrokontrolér pracoval jako USB hostitel i host. Dokonce je možné při použití speciálního AB konektoru, aby pracoval jako hostitel a host zároveň. Můj modul potřebuje ale jen připojit flash paměť, takže mikrokontrolér pracuje jen v hostitelském módu a pro zjednodušení jsem v nastavení knihovny povolil připojit jen 1 zařízení, takže není možné jich pomocí USB rozbočovače připojit více.

Knihovna obstarává veškerou práci s médiem a programátorovi jsou k dispozici funkce ze standardní knihovny pro práci se soubory (`fopen, fread, fwrite...`). V základním

nastavení knihovny ale mohou být otevřeny jen 2 soubory a struktura s údaji o otevřeném souboru jsou uloženy v paměti flash. Aby bylo možné mít otevřené všechny soubory z playlistu, musel jsem knihovnu upravit tak, aby bylo možné otevírat více souborů a aby se údaje ukládaly do paměti RAM, které je dostatek.

5.3 Pevný disk

Načítání dat z pevného disku se mně nakonec nepodařilo implementovat. Použil jsem řadič a knihovnu od mého spolužáka Bc. Stanislava Sigmunda, který je vytvořil v rámci své bakalářské práce[18]. Po inicializaci řadiče a pokusu načíst informace o disku, inicializační funkce vrátí chybu že disk je vadný, což ale nejspíš není pravda. Bohužel už ale nemám čas zjišťovat, kde konkrétně je chyba, a tak připojení pevného disku nechávám do budoucna jako další možné rozšíření.

5.4 Grafická knihovna

K tvorbě a vykreslování uživatelského prostředí jsem zvolil knihovnu od firmy Microchip[9], která je za podmínky použití s mikrokontroléry této firmy dostupná zdarma. Další možností by byla grafická knihovna firmy Ramtex, která ale není zdarma ani pro výukové účely.

Knihovna firmy Microchip umožňuje vytvářet běžné objekty, jaké známe z prostředí operačních systémů na PC, i přímo kreslit jednoduché tvary na displej.

5.4.1 Low-level ovladač displeje

Knihovna obsahuje ovladače pro některé typy řadičů displeje, ale ovladač pro řadič, který je použit, chybí. K jeho naprogramování jsem použil šablonu, která se nacházela u ostatních ovladačů v knihovně.

Bylo nutné vytvořit následující funkce, které jsou vyžadovány vyššími vrstvami grafické knihovny:

- `void ResetDevice(void)`
  - Nastaví 16-ti bitový port mikrokontroléru pro komunikaci s řadičem displeje (zejména vypne adresovou sběrnici a nastaví mód, ve kterém se používají signály *read/write* a *enable strobe*).
  - Resetuje řadič displeje a vypne jeho (po resetu implicitní) stand-by mód.
  - Nastaví řadič displeje tak, aby komunikoval pomocí 16-ti bitové sběrnice, a deaktivuje jeho RGB rozhraní.
  - Nastaví zobrazování jen 65536 barev a automatickou inkrementaci adresy při zápisu do paměti řadiče podle toho, jestli chceme s displejem pracovat na výšku nebo na šířku.
- `void PutPixel(SHORT x, SHORT y)`
  - Nastaví pixel na souřadnicích `x`, `y` na barvu danou globální proměnnou `_color`.
- `WORD GetPixel(SHORT x, SHORT y)`
  - Vrátí barvu pixelu na souřadnicích `x`, `y`.

- `void Bar(SHORT left, SHORT top, SHORT right, SHORT bottom)`
  - Vyplní oblast, která je specifikována parametry funkce, barvou danou globální proměnnou `_color`.

Volitelně je možné ještě doplnit funkce pro vykreslování pixmap s různou hloubkou barev, ale to by bylo výhodné jen tehdy, pokud by bylo možné jejich vykreslení nějakým způsobem optimalizovat přímo pro tento řadič. To ale není možné a tak jsou k jejich vykreslování použity funkce z vyšší vrstvy knihovny.

5.4.2 Ovladač touch-screenu

Grafická knihovna již obsahuje kód pro zjišťování místa dotyku na displeji. Bylo třeba jen upravit názvy registrů, kam jsou připojeny vývody dotykové vrstvy, a zadat kalibrační údaje, které se rovnají minimálním a maximálním úrovním napětí ve vodorovném a svislém směru. Tyto údaje je možné mít uložené přímo v zdrojovém kódu nebo je získat za běhu aplikace spuštěním kalibrační funkce, která je uloží do EEPROM paměti. Tu ale nahrazuje u procesoru PIC32 flash paměť, která je při každém přeprogramování přepsána. Proto jsem zvolil při vývoji výhodnější uložení kalibračních dat přímo do zdrojového kódu (za celou dobu vývoje nebylo třeba tyto data měnit), ale při ostrém nasazení by bylo vhodné umožnit uživateli rekalibraci.

Odporový touch screen se připojuje 4 vodiči k A/D převodníkům mikrokontroléru. Tyto vodiče jsou označeny X+, X-, Y+ a Y-. Postup získání souřadnic dotyku je následující:

1. Na vodič Y+ se připojí napájecí napětí a vodič Y- se připojí k zemi. Piny mikrokontroléru, kam jsou připojeny vodiče X+ a X-, jsou ve stavu vysoké impedance. Toto je klidový stav.

2. Na vodiči X+ nebo X- se pomocí A/D převodníku zjistí napětí, které je přímo úměrné pozici dotyku na svislé ose displeje.

3. Piny, kde jsou připojeny vodiče Y+ a Y-, přejdou do stavu vysoké impedance, vodič X+ se připojí na napájecí napětí a vodič X- na zem.

4. Na vodiči Y+ nebo Y- lze nyní odečíst napětí, které je přímo úměrné místu dotyku na vodorovné ose displeje.

5. Získané úrovně napětí převedeme použitím kalibračních dat na souřadnice v pixelech.

$$x = \frac{width_{LCD} \cdot (X_{A/D} - X_{min})}{X_{max} - X_{min}}$$

$$y = \frac{height_{LCD} \cdot (Y_{A/D} - Y_{min})}{Y_{max} - Y_{min}}$$

Hardware modulu sice umožňuje generovat signál přerušení při dotyku na displeji, ale knihovna používá časovač T3, pomocí kterého se každých 100 μs zjišťuje, zda nedošlo k dotyku. Protože tímto způsobem lze, narozdíl od přerušení, detekovat i změnu místa dotyku (posun po displeji), použil jsem funkci knihovny bez externího signálu přerušení.

Při zjištění dotyku, uvolnění nebo pohybu po displeji je generována událost, která je později zpracována v hlavní smyčce programu (viz. 5.4.6).

5.4.3 Nastavení grafické knihovny

Veškerá nastavení grafické knihovny se provádějí v souboru `GraphicsConfig.h` pomocí klíčového slova `#define`. Nastavit lze rozměry, barevnou hloubku, orientaci a ovladač displeje. Pokud je použit vlastní ovladač, musíme ještě upravit soubory `Graphics/Graphics.h` a `Graphics/DisplayDriver.h`, kde je třeba doplnit definici a cestu k našemu low-level ovladači.

Dále je nutné v tomto souboru odkomentovat jednotlivé použité komponenty a komponenty, které nejsou použity, zakomentovat. Jinak by totiž byly přilinkovány a zbytečně by zabíraly paměť v mikrokontroléru.

Je možné volit i mezi blokujícím a neblokujícím vykreslováním komponent. Při neblokujícím vykreslování, narozdíl od blokujícího, jsou komponenty vykreslovány v několika fázích, mezi nimiž vždy proběhne hlavní smyčka programu, takže je možné v průběhu vykreslování složité komponenty provádět i jiné operace.

5.4.4 Vytváření komponent

Knihovna umožňuje vytvářet tyto komponenty: Button, Chart, Window, CheckBox, RadioButton, EditBox, ListBox, Slider/ScrollBar, ProgressBar, StaticText, Picture, GroupBox, RoundDial, Meter a TextEntry. Z nich používám jen některé (Button, ListBox, Slider/ScrollBar a StaticText), protože každá další komponenta zabere místo ve flash paměti a té je velký nedostatek, protože použitý kompilátor má v akademické verzi omezení, kdy je zakázáno použití optimalizací, které by přeložený kód zmenšily. K dalšímu vývoji doporučuji vyměnit mikrokontrolér za typ s větší flash pamětí.

Komponenty je možné vytvořit všechny najednou v inicializační části programu nebo je vytvářet postupně za běhu až podle potřeby před vykreslením. První možnost má výhodu v možnosti nastavování parametrů i když komponenta právě není vykreslena, ale je potřeba více paměti a s vykreslováním komponent je složitější práce, protože je třeba nastavovat, která komponenta má být vykreslena a která skryta. Výhody a nevýhody vytváření komponent za běhu jsou právě opačné k vytvoření všech komponent v inicializační části.

Vzhledem k tomu, že uživatelské prostředí používá záložky, mezi kterými uživatel přepíná, je nastavování parametrů právě skrytým komponentám nutností, takže všechny komponenty vytvářím v inicializační části aplikace.

Každá komponenta je tvořena svojí strukturou, kde jsou uchovávány její vlastnosti a stavy. Součástí této struktury je hlavička, která je stejná pro všechny komponenty, a jsou v ní uloženy rozměry a umístění komponenty, její typ, stav (skrytá, neaktivní ...) a jedinečné identifikační číslo, pomocí kterého lze kdekoliv v programu získat ukazatel na tuto komponentu.

Struktura komponenty se vytváří funkcí *zkratka_komponenty*`Create`, jejíž přesný název a parametry lze nalézt v dokumentaci grafické knihovny[9] nebo na přiloženém CD. Této funkci se mimo jiné předává ukazatel na grafické schéma, což je struktura, která určuje barvy, jakými se komponenta vykreslí. Změnou tohoto schéma tak lze změnit vzhled více komponent.

5.4.5 Vykreslování komponent

Komponenty se vykreslují při zavolání funkce `WORD GOLDraw()` a to jen ty, které mají nastaven stavový bit pro překreslení. Tato funkce by měla být volána v hlavní smyčce programu, protože při neblokujícím nastavení je překreslování rozděleno do několika fází a

po každé fázi funkce skončí (vrací hodnotu 0). Po poslední fázi vrátí nenulovou hodnotu a až potom by se měly volat funkce pro obsluhu událostí.

Zobrazení spektra přehrávaného signálu

Spektrum signálu je vykreslováno jednou za 1000 cyklů hlavní smyčky, což je přibližně 25× za vteřinu. Po načtení dat z audio kodeku (kap. 5.1.3) dostáváme 22 5-ti bitových hodnot. Abych se vyhnul přepočítávání hodnot na pixely, stanovil jsem maximální výšku sloupců na 31 pixelů.

Nejdříve je přepsána podkladovou barvou ta část sloupce, která nebude zobrazena, potom je vykreslen sloupec s výškou rovnou získané hodnotě pro dané pásmo. Nakonec je vykreslena červená čára, která reprezentuje nejvyšší hodnotu.

5.4.6 Zpracování událostí

Grafická knihovna vyžaduje po uživateli implementaci dvou callback funkcí:

- Funkce `WORD GOLMsgCallback(WORD objMsg, OBJ_HEADER* pObj, GOL_MSG* pMsg)`, která je volána vždy, když nějaká komponenta vyvolá událost. Funkci je předán ukazatel na hlavičku komponenty (lze přetypovat na ukazatel na konkrétní komponentu, protože ve struktuře komponenty je ukazatel na její hlavičku na prvním místě a ukazatele jsou tedy stejné) a přeložená (`objMsg`) a nepřeložená (`pMsg`) zpráva, která obsahuje údaje o události.

  V mojí implementaci této funkce nejdříve zjišťuji druh události a až poté jejího původce, což je trochu nepřirozené, ale lze tak napsat kratší a úhlednější kód, protože např. u tlačítka zpracovávám jen událost stisku.

  Pokud funkce vrátí nenulovou hodnotu, je ještě provedena standardní procedura, např. při stisku tlačítka změna jeho vzhledu.

- Funkce `WORD GOLDrawCallback()`, která je volána vždy po vykreslení komponent a je zde možné vykreslit dodatečné úpravy komponent nebo komponenty zcela vlastní. Tuto funkci ale nijak nevyužívám.

5.5 Uživatelské rozhraní

Uživatelské rozhraní je rozděleno na záložky, mezi nimiž lze přepínat tlačítko na pravé straně displeje. Jejich označení je vysvětleno na obrázku 5.4.

Na všech záložkách jsou také ve spodní části umístěny tlačítka ovládající přehrávání a umožňující skok na předcházející nebo následující skladbu v seznamu skladeb.

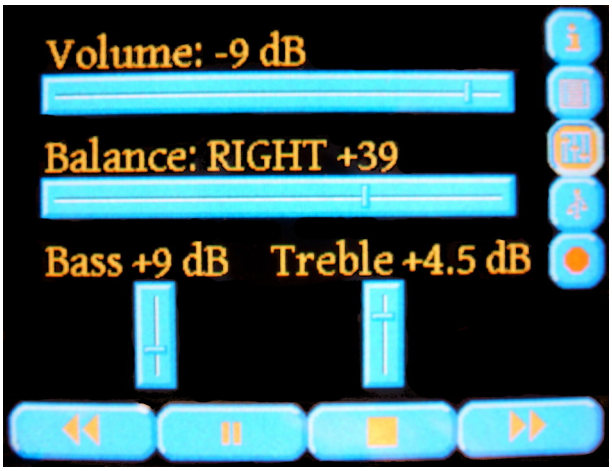
5.5.1 Informace o přehrávané skladbě

Informace o skladbě se načítají při otevření souboru z ID3 tagů MP3 souboru. Načítání informací z jiných formátů není podporováno. Podporovány jsou jen ID3 tagy verze 1, které mají pevnou strukturu a nachází se na konci souboru. ID3 tagy verze 2 nejsou podporovány, protože mají proměnlivou velikost a ta může být až v řádu megabytů, a to by nebylo možné v mikrokontroléru zpracovat. České znaky také nejsou podporovány, protože by bylo potřeba vytvořit nový font.

Obrázek 5.4: Ukázka zobrazení informací o skladbě

Na záložce s informacemi je zobrazováno jméno interpreta, název skladby, název alba, rok vydání, číslo stopy a žánr právě přehrávané skladby. Ve spodní části je v průběhu přehrávání zobrazováno aktuální frekvenční spektrum.

5.5.2 Ovládání hlasitosti, vyvážení, basů a výšek



Obrázek 5.5: Ukázka nastavení přehrávání

Pomocí vodorovných posuvníků lze nastavit hlasitost a vyvážení pravého a levého kanálu. Kodeku se předává hlasitost zvlášť pro pravý a levý kanál a při nastavování se provádí přepočet (viz. kap. 5.1.4). Posuvník pro vyvážení je nastavitelný v intervalu $\langle -100; +100 \rangle$.

Svislými posuvníky je možné upravit basy a výšky výstupního signálu.

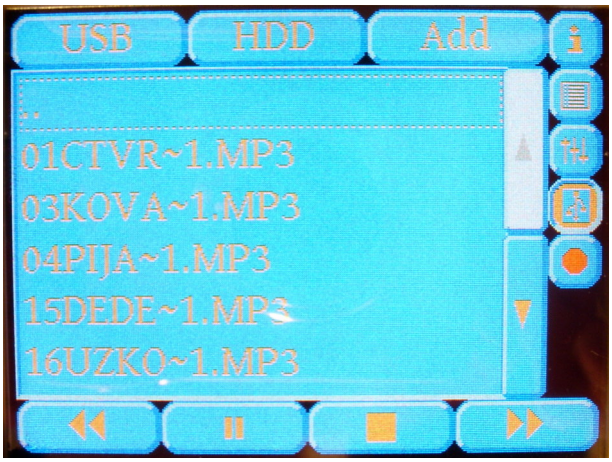
Obrázek 5.6: Ukázka seznamu skladeb

5.5.3 Seznam skladeb

Seznam skladeb obsahuje jména interpretů a názvy skladeb, které byly již přehrány a které se teprve přehrávat budou. Aktuálně přehrávaná skladba je zvýrazněna. Skladby lze mazat a měnit jejich pořadí. Z obvyklých funkcí jen není implementováno opakování a náhodný výběr skladeb.

Přidávání skladeb do seznamu se provádí přes záložku, která je popsána v následující kapitole 5.5.4.

5.5.4 Procházení souborů na médiu



Obrázek 5.7: Ukázka seznamu souborů

Tuto záložku jsem vytvořil, aby bylo možné procházet soubory na médiu. V horní části je možné zvolit mezi USB médiem a pevným diskem, který se mi ale nepodařilo zprovoznit a tak je tlačítko neaktivní.

K získání seznamu souborů v adresáři používám funkce `FindFirst` a `FindNext` z USB

knihovny, kterým předávám atributy, které mají nalezené soubory splňovat. Lze tak získat např. jen adresáře nebo jen soubory s příponou *.mp3. Funkce ale vrací seznam neseřazený, proto jej pro větší uživatelský komfort řadím podle abecedy – nejdříve adresáře a potom soubory. Také se zobrazují jen soubory, které lze přehrát.

Šipkami po pravé straně seznamu se lze v seznamu pohybovat. Přidání souboru do seznamu skladeb se provede klepnutím na jeho název a následně na tlačítko Add.

Názvy souborů jsou v krátkém DOS formátu, protože USB knihovna dlouhé názvy nepodporuje.

Kapitola 6

Další možná rozšíření

Modul by dále bylo možné vylepšit těmito rozšířeními:

- Vyměnit mikrokontrolér za typ s větší flash pamětí, protože se ukázalo, že 128 kB je málo. Hodně paměti totiž zaberou použité knihovny a ke konci vývoje jsem měl paměť zaplněnou z více než 95 %. Pravděpodobně by ani nebylo možné použít softwarovou knihovnu pro práci s pevným diskem i kdyby se mi jej podařilo zprovoznit, protože by se už nevešla do paměti mikrokontroléru.

  Vyrábí se mikrokontroléry s až 512 kB flash paměti, konkrétně typ PIC32MX460F512L by byl vhodný. Rozložení vývodů je u těchto typů procesorů stejné.

- Zprovoznit načítání a ukládání souborů na pevný disk.

- Implementovat nahrávání zvuku do formátu Vorbis OGG. K tomu je potřeba nahrát do audio kodeku plugin, který toto umožní. Je ale potřeba vyměnit mikrokontrolér za typ s větší pamětí, protože nyní pro plugin není v mikrokontroléru dostatek paměti.

- Vytvořit font, který bude podporovat české znaky. Také by bylo vhodné vytvořit menší font než je dodávaný s knihovnou, protože nyní se na displej mnoho textu nevejde.

Závěr

Zadání mé diplomové práce mně určuje vytvořit modul k FITkitu, který umožní přehrávat soubory MP3. Toto zadání jsem se rozhodl značně rozšířit o přehrávání dalším formátů, ovládání pomocí dotykového LCD displeje, pevný disk a USB rozhraní.

Zadání jsem splnil celé a podařilo se mně zprovoznit i většinu mých rozšíření, pouze nelze přehrávat Windows Media Audio soubory a není možné komunikovat s pevným diskem. Tyto nedostatky by nejspíš nebyl problém odstranit, ale z důvodu časové tísně, jsem se jimi hlouběji nezabýval.

Vlastní přínos shledávám v prohloubení mých znalostí v oblastech návrhu vestavěných systémů a plošných spojů a v hledání příčin chyb, kterých jsem se v průběhu vývoje dopustil.

Praktické využití modulu vidím v dalším rozvoji platformy FITkit. Zejména dotykový LCD displej má široké možnosti uplatnění. Do budoucna si myslím, že by bylo vhodné navrhnout univerzální rozhraní, které by umožňovalo připojit k FITkitu více takovýchto rozšiřujících modulů.

Literatura

[1] Ogg Vorbis Documentation. [online], [cit. 21. 5. 2009].
URL http://www.xiph.org/vorbis/doc/

[2] Bartoš, P.: Generátor funkcí pomocí D/A převodníku FITkitu. Bakalářská práce. FIT VUT Brno. 2007.

[3] Chandraiah P., Dömer R.: Specification and Design of a MP3 Audio Decoder. Technická zpráva, Center for Embedded Computer Systems, University of California, Irvine, 5 2005, [online].
URL www.cecs.uci.edu/technical_report/TR05-04.pdf

[4] Densitron displays: Preliminary Product Specification - C0283QGLH-T. [online], version 1.7, [rev. 18. 12. 2007], [cit. 30. 12. 2008].
URL http://www.farnell.com/datasheets/116930.pdf

[5] Drábek V., Herout A.: Kódování audia. [online], [cit. 28. 12. 2008].
URL https://www.fit.vutbr.cz/study/courses/GMU/private/lect/GMP13-MPEG1A.pdf

[6] Fairchild semiconductor: LM317L. [online], 3 2002, [cit. 20. 5. 2009].
URL www.fairchildsemi.com/ds/LM%2FLM317L.pdf

[7] Markovič, J.: Firmware / Sériové rozhraní. [online], [cit. 20. 5. 2009].
URL http://web.archive.org/web/20071021092456/merlin.fit.vutbr.cz/FITkit/docs/firmware/20060414ser.html

[8] Martínek, T.: Technologie FPGA. [online], [cit. 6. 5. 2007].
URL https://www.fit.vutbr.cz/study/courses/PCS/private/prednasky/03_technologie_fpga/fpga_techn.pdf

[9] Microchip: Graphic Design Resources. [online], [cit. 30. 12. 2008].
URL http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE\&nodeId=2608\&page=1\&param=en532061\&redirects=Graphics

[10] Microchip: PIC32MX3XX/4XX Family Data Sheet. [online], 1 2008, [cit. 20. 5. 2009].
URL http://ww1.microchip.com/downloads/en/DeviceDoc/61143E.pdf

[11] Pech, J.: Programovatelné logické obvody. [online], [cit. 6. 5. 2007].
URL http://www.sweb.cz/fpga/

[12] Přispěvatelé Wikipedie: Codebook. [online], [cit. 21. 5. 2009].
URL http://en.wikipedia.org/wiki/Codebook

[13] Přispěvatelé Wikipedie: Modified discrete cosine transform. [online], [cit. 28. 12. 2008].
URL http://en.wikipedia.org/wiki/Modified_discrete_cosine_transform

[14] Přispěvatelé Wikipedie: MP3. [online], [cit. 28. 12. 2008].
URL http://cs.wikipedia.org/wiki/Mp3

[15] Přispěvatelé Wikipedie: MPEG-1. [online], [cit. 28. 12. 2008].
URL http://en.wikipedia.org/wiki/MPEG-1

[16] Přispěvatelé Wikipedie: Pulzně kódová modulace. [online], [cit. 28. 12. 2008].
URL http://cs.wikipedia.org/w/index.php?title=Pulzn%C4%9B_k%C3%B3dov%C3%A1_modulace\&oldid=3327222

[17] Samsung electronics: S6E63D6 - datasheet. [online], version 1.10, [rev. 10. 9. 2007], [cit. 28. 12. 2008].
URL http://data.4dsystems.com.au/downloads/micro-OLED/uOLED-320XX-PMD3/Docs/Pdf/S6E63D6X_REV1.10.pdf

[18] Sigmund, S.: Rozhraní IDE pro platformu FITkit. Bakalářská práce. FIT VUT Brno. 2007, [cit. 31. 12. 2008].
URL http://www.fit.vutbr.cz/study/DP/rpfile.php?id=5519

[19] Solution, V.: VS10xx Spectrum Analyzer. [online], 6 2008, [cit. 20. 5. 2009].
URL http://www.vlsi.fi/fileadmin/software/VS10XX/spectrumAnalyzer.pdf

[20] Texas Instruments: TPS65136 datasheet. [online], 4 2008, [cit. 30. 12. 2008].
URL http://focus.ti.com/lit/ds/symlink/tps65136.pdf

[21] Toman, M.: O MP3. [online], Západočeská univerzita v Plzni, 29. 9. 2001, [cit. 26. 12. 2008].
URL http://home.zcu.cz/~mtoman/mp3.pdf

[22] Vašíček Z., S. K.: Firmware / Komunikační systém. [online], [cit. 20. 5. 2009].
URL
http://merlin.fit.vutbr.cz/FITkit/docs/firmware/fpga_interconect.html

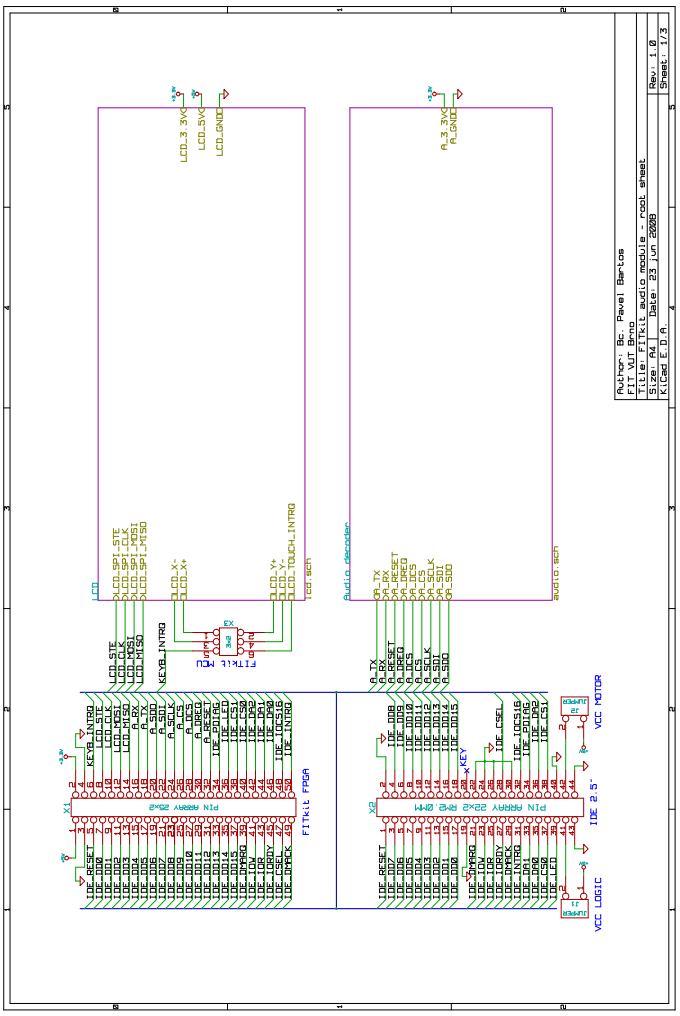
[23] VLSI Solution: VS1053b datasheet. [online], version 1.01, [rev. 22. 5. 2008], [cit. 2. 1. 2009].
URL http://www.vlsi.fi/fileadmin/datasheets/vlsi/vs1053.pdf

[24] Zemčík, P.: Komprese zvuku. [online], [cit. 28. 12. 2008].
URL https://www.fit.vutbr.cz/study/courses/MUL/private/download/MUM%20mp3.pdf

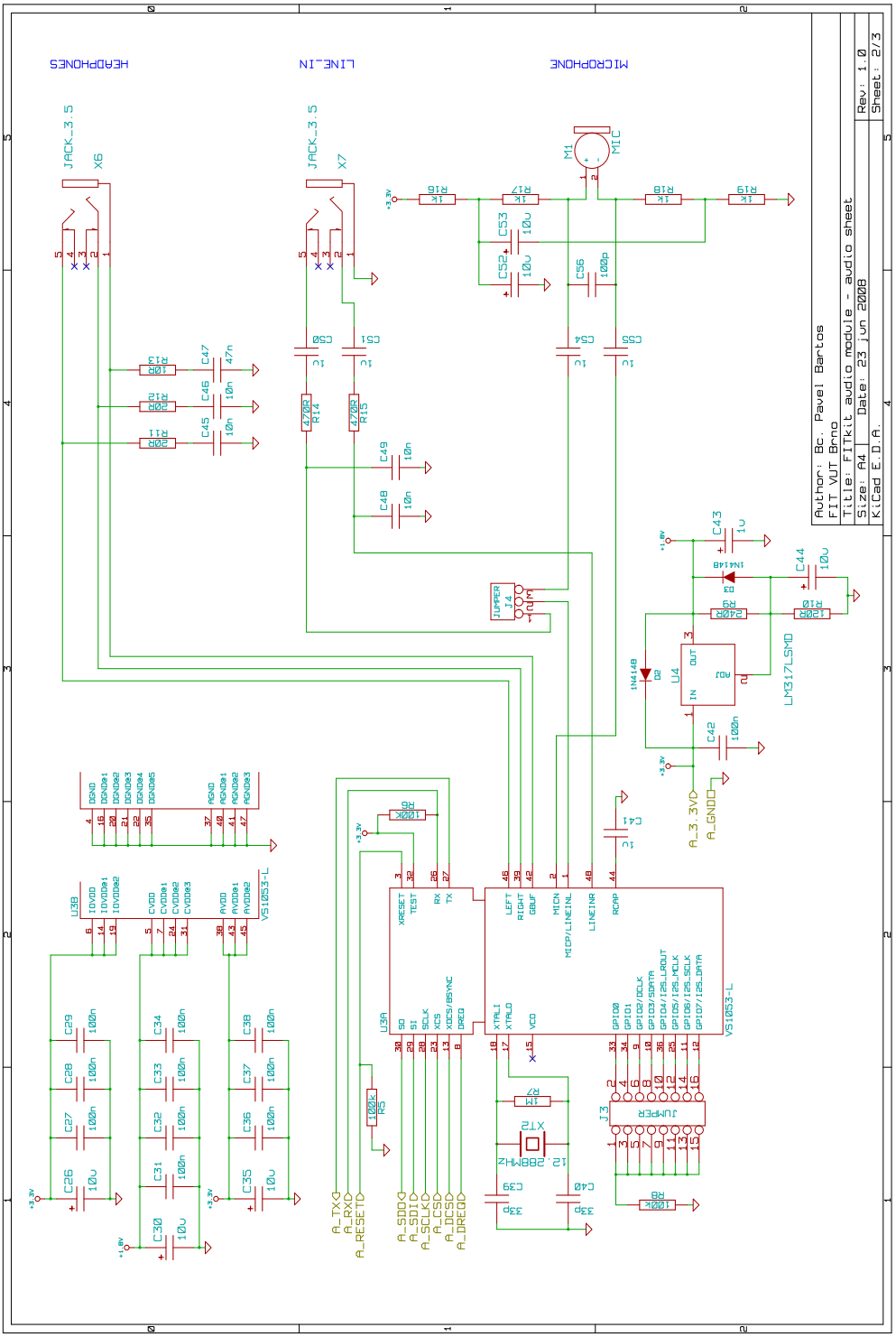
Seznam příloh

Příloha A

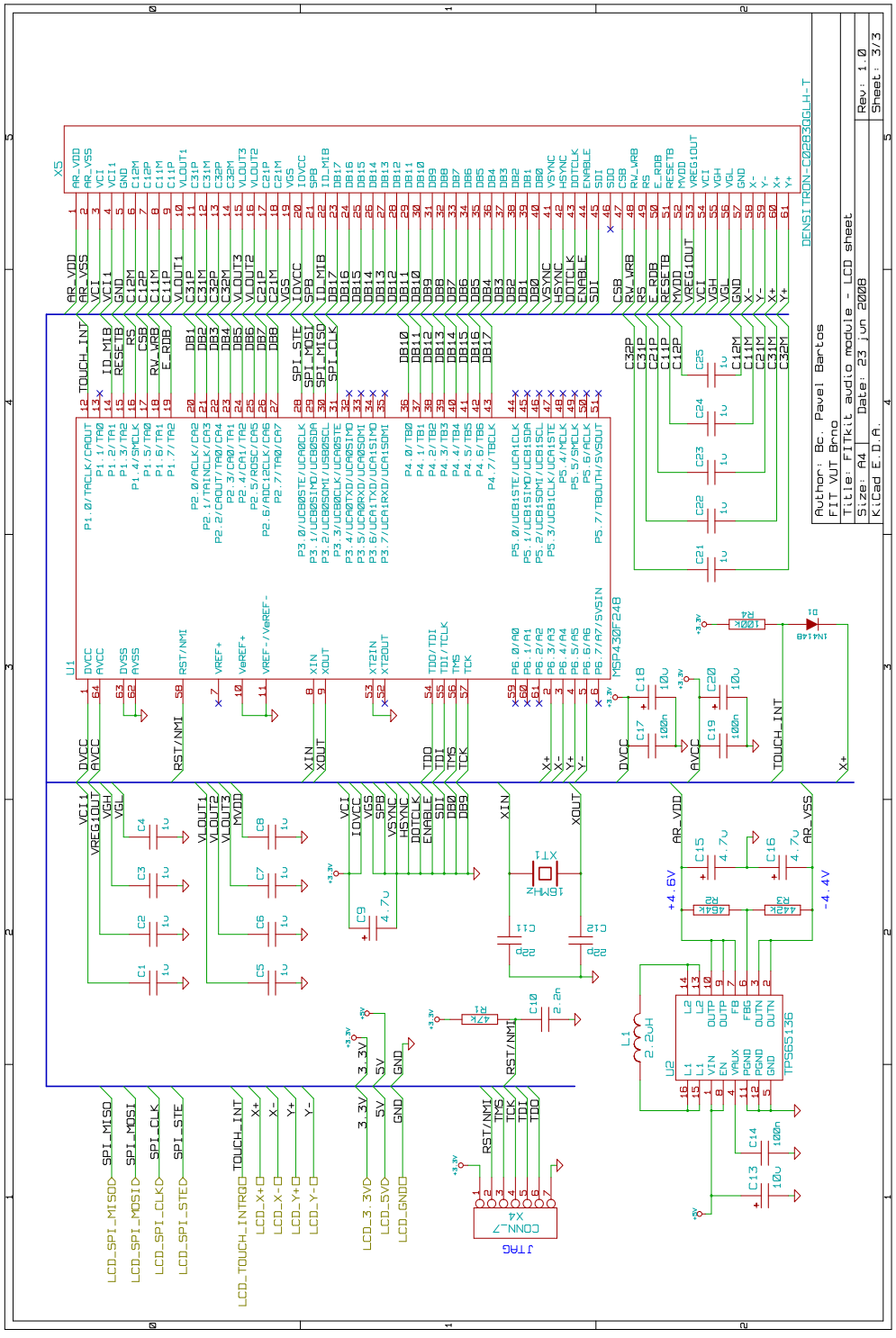
Elektrotechnické schéma – 1. verze



Obrázek A.1: Elektrotechnické schéma celého modulu



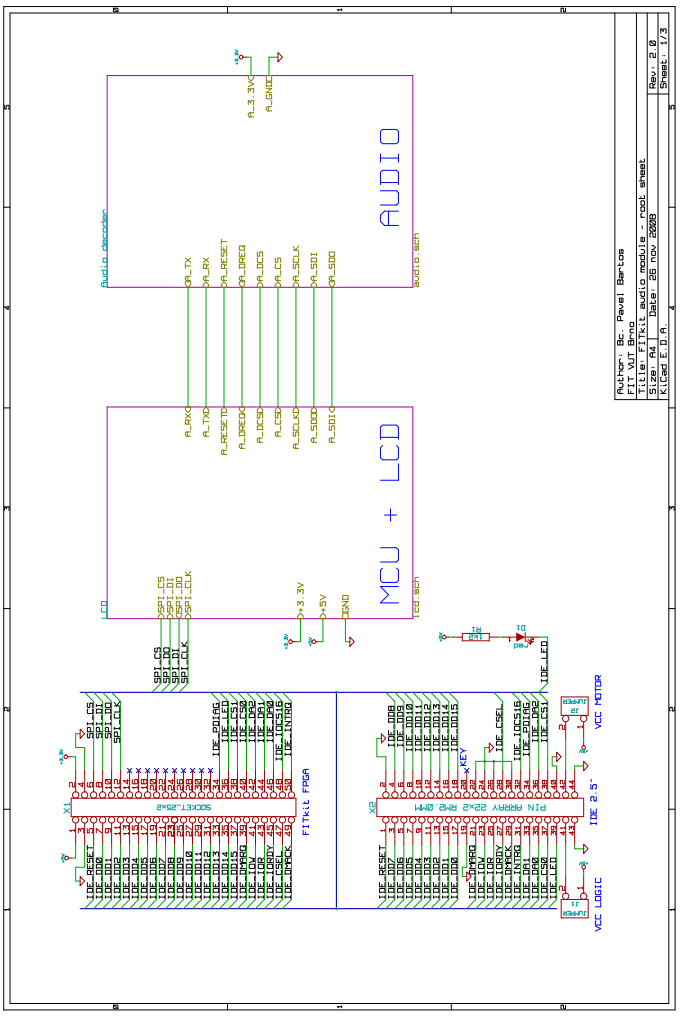
Obrázek A.2: Elektrotechnické schéma audio části



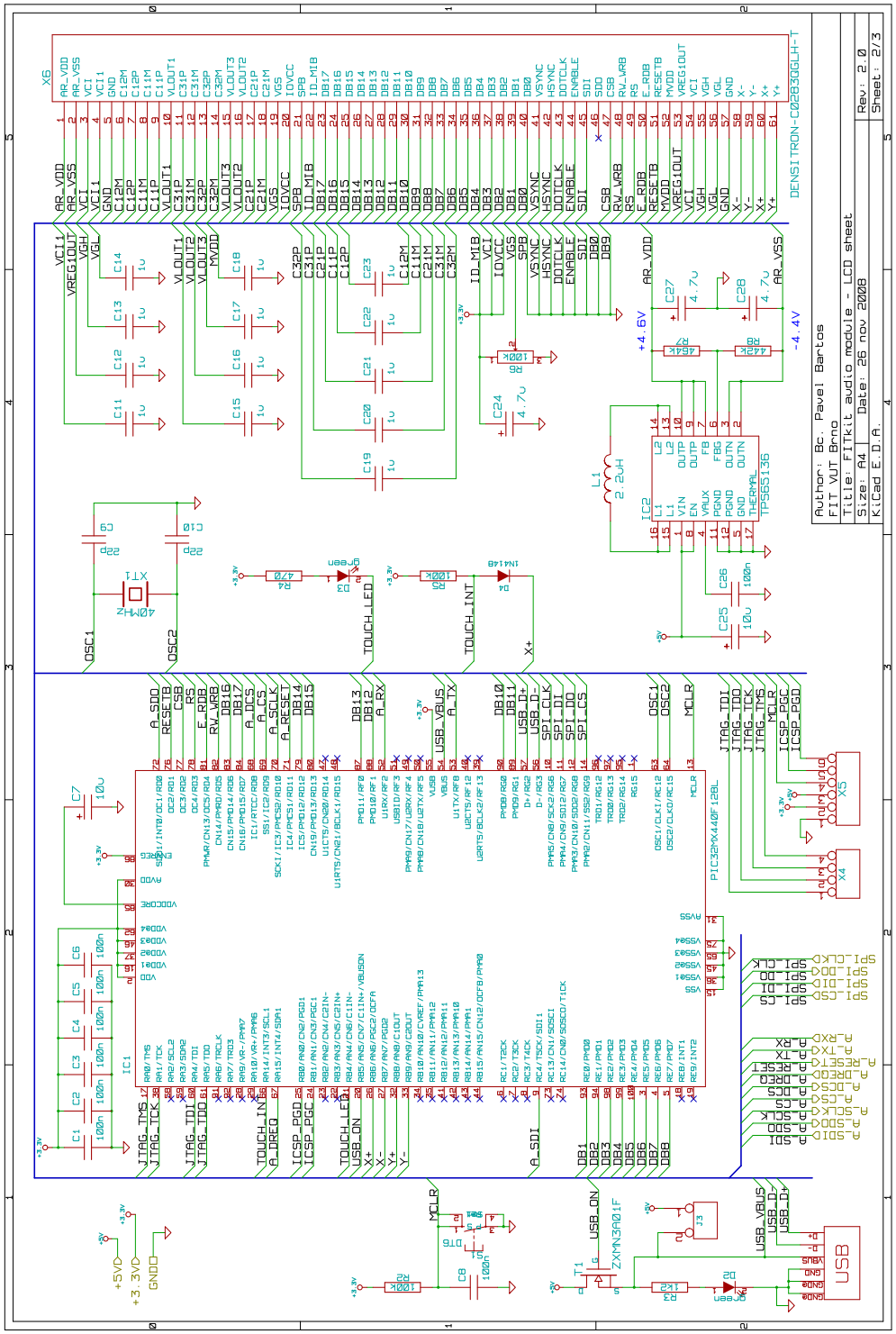
Obrázek A.3: Elektrotechnické schéma LCD části

Příloha B

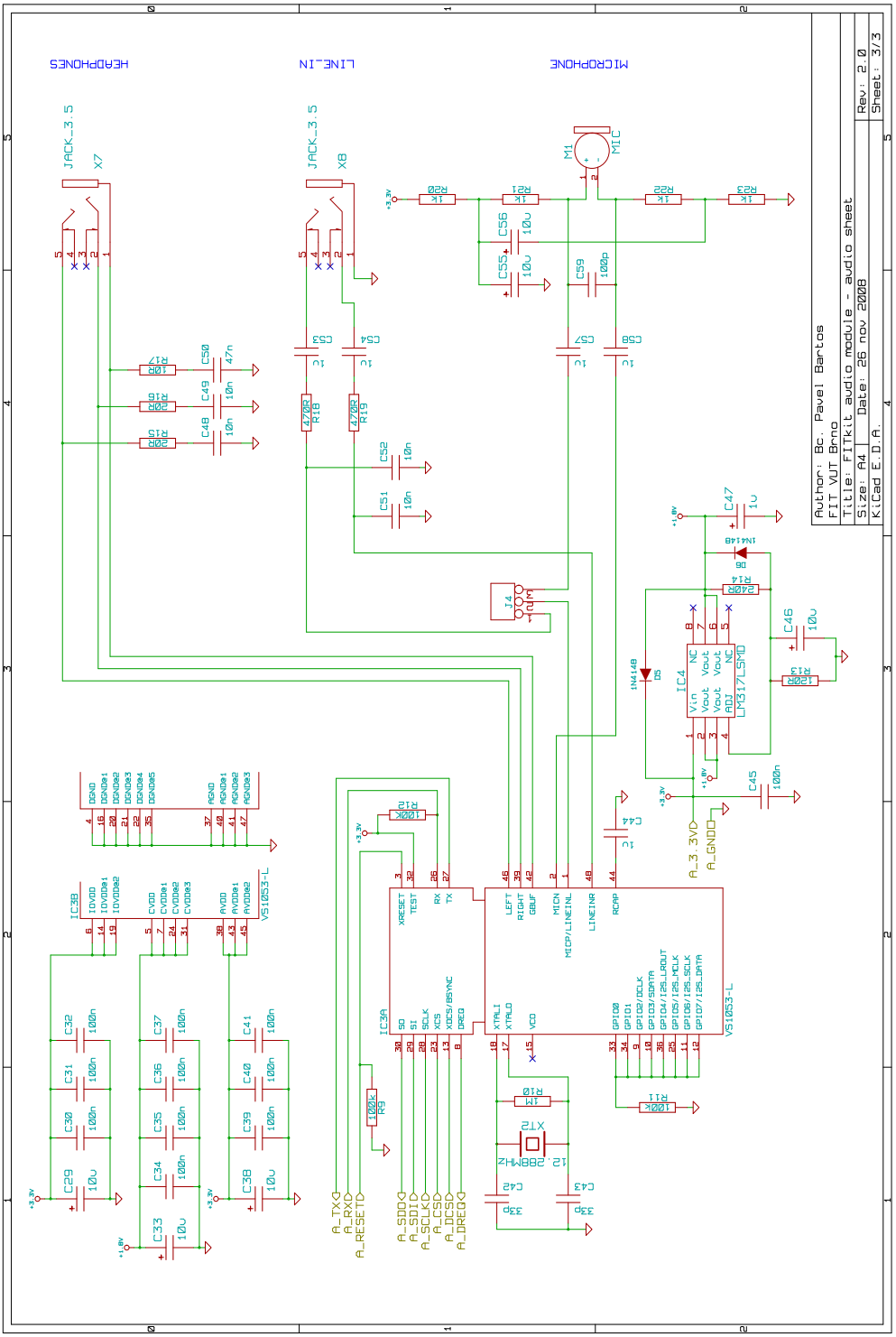
Elektrotechnické schéma – 2. verze



Obrázek B.1: Elektrotechnické schéma celého modulu



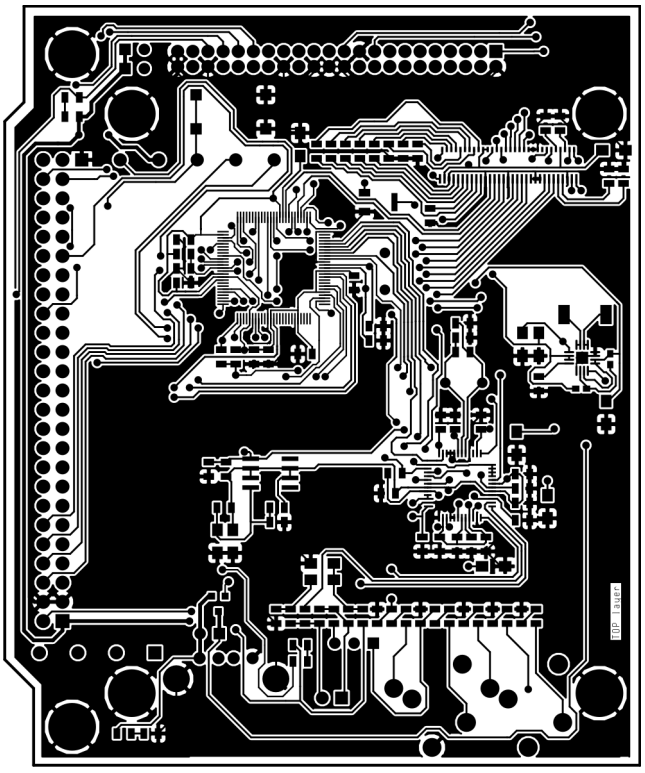
Obrázek B.2: Elektrotechnické schéma LCD části



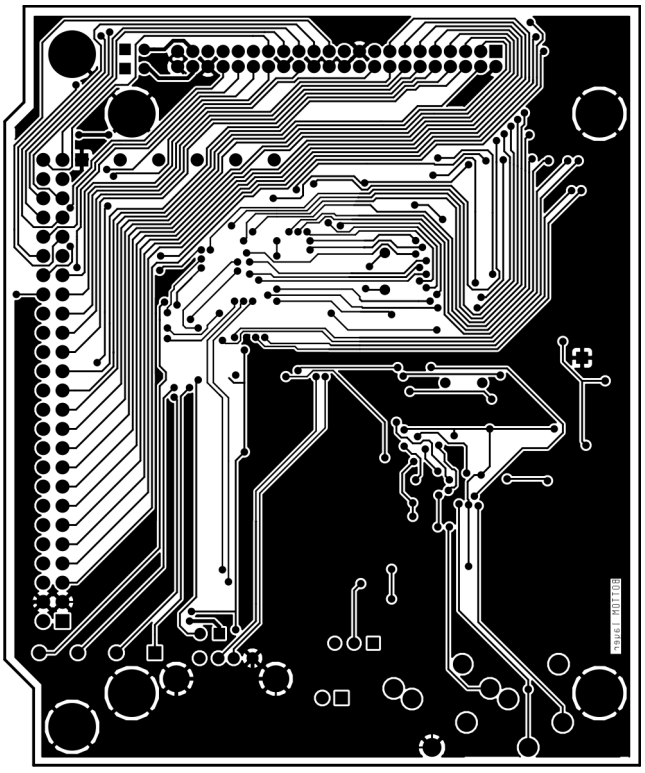
Obrázek B.3: Elektrotechnické schéma audio části

Příloha C

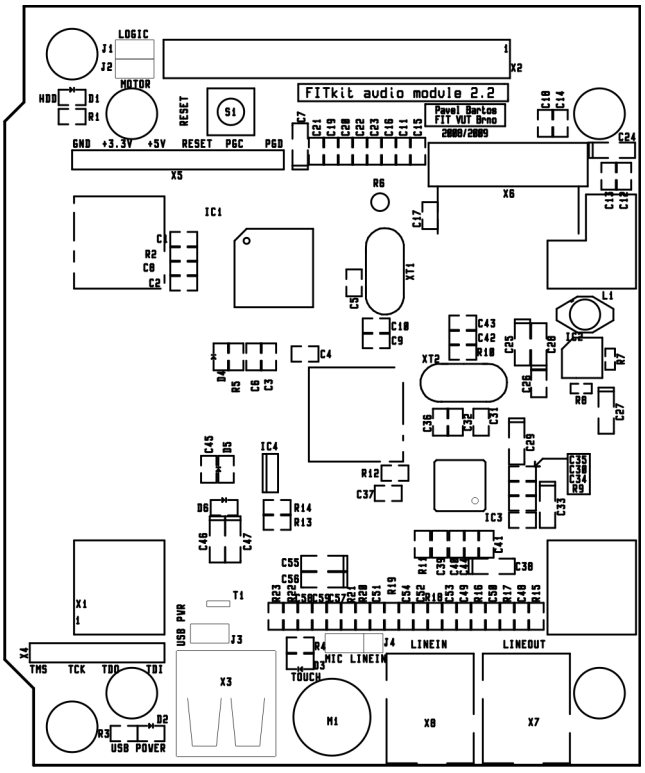
Obraz desky plošných spojů



Obrázek C.1: Deska plošných spojů – strana součástek



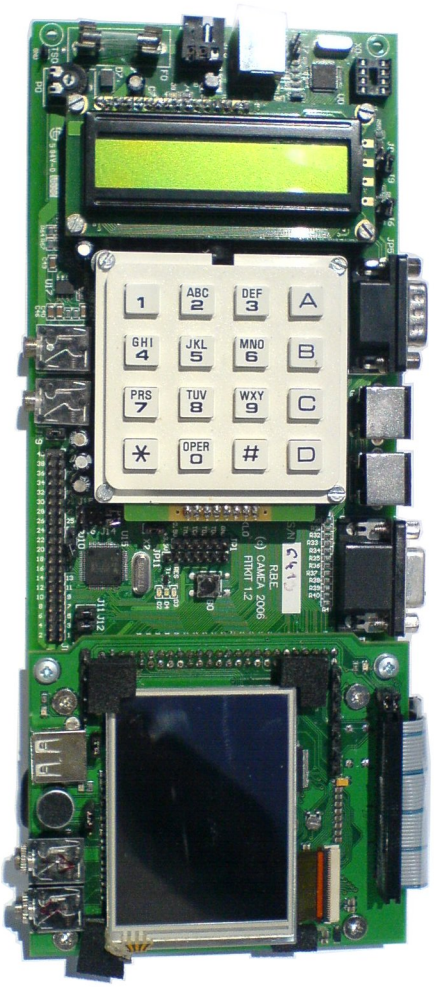
Obrázek C.2: Deska plošných spojů – spodní strana



Obrázek C.3: Deska plošných spojů – servisní potisk

Příloha D

Fotografie FITkitu s připojeným modulem



Obrázek D.1: FITkit s modulem